修 士 論 文

# $L_1$ノルムを用いた点列の一連節折れ線近似アルゴリズムの開発に関する研究

北陸先端科学技術大学院大学
情報科学研究科情報処理学専攻

角川 肇喜

2005年3月

修 士 論 文

# $L_1$ノルムを用いた点列の一連節折れ線近似アルゴリズムの開発に関する研究


指導教官　浅野 哲夫 教授

審査委員主査　浅野 哲夫 教授
審査委員　平石 邦彦 教授
審査委員　宮地 充子 助教授

北陸先端科学技術大学院大学
情報科学研究科情報処理学専攻

310068 角川 肇喜



提出年月: 2005 年 2 月

## 概 要

$L_1$ ノルムを用いて点列の折れ線近似問題を効率的に解く実用的なアルゴリズムは現時点においても開発されていない．また，$L_1$ 折れ線近似問題の基礎となる屈折点が 1 つの $L_1$ 一連節折れ線近似問題の最適解を求めるアルゴリズムは既に考案されているが，計算時間の点で実用的ではない．そこで，本研究では $L_1$ 一連節折れ線近似問題を効率的に解く実用的なアルゴリズムの開発を行った．折れ線近似問題の難しさは屈折点の数と位置の決定にある．本研究の提案手法では，はじめに入力の点列を屈折点 1 つの折れ線で近似するのが相応しいかの判定を行い，相応しいと判定されたものに対してのみ $L_1$ 一連節折れ線近似を行う事とした．また，屈折点の位置の決定に関しては 2 通りの方法を提案する．開発したアルゴリズムの評価は，それを実装し点列に対して実行した結果を，最適な折れ線近似の結果と比較することにより行った．

# 目 次

# 第1章　序論

## 1.1　はじめに

はじめに，本論文で用いるいくつかの用語の解説を行う．

### 1.1.1　直線近似

直線近似とは図 1.1 に示す様に，平面上に与えられた点集合（点列）に，それを近似するような直線を当てはめることを言う．直線の当てはめとは，具体的には直線の傾き $a$ と切片 $b$ を決定することである．実際には，実験データ等の二変数間の相関を求める問題などが有名で，統計学の分野では，直線回帰または単回帰とよばれる，回帰分析問題の一つである．

以降の本論文における直線近似とは，次のような場合の点列への直線の当てはめを指す．$x\_y$ 平面上に $n$ 個の点 $p_1 = (x_1, y_1), p_2 = (x_2, y_2), \ldots, p_n = (x_n, y_n)$ からなる点集合 $S$，が与えられたとき，点集合 $S$ をより良く近似する直線 $y = ax + b$ を当てはめる．当てはめた直線 $y = ax + b$ を近似直線という．また，直線近似を求める問題を直線近似問題，それを解くアルゴリズムを直線近似アルゴリズムと呼ぶ．

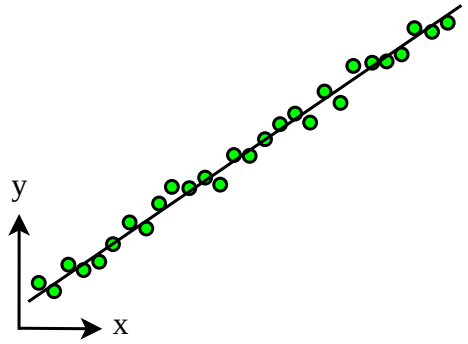



図 1.1: 点列の直線近似

### 1.1.2 折れ線近似

折れ線近似とは，図 1.2 に示すように $x\_y$ 平面上に与えられた点集合に，それを近似するように $x$ 単調な，部分的に線形かつ連続である折れ線 $Q$ を当てはめることである．部分的に線形かつ連続の意味とは，つまり，屈折点が $k$ 個の折れ線の場合，両端の 2 つの半直線と残り $k-1$ 個の直線分，計 $k+1$ 個の直線成分が，隣接するもの同士でそれぞれ端点を共有しながら折れ線を構成していることを指す．また，そのことから屈折点が $k$ 個の折れ線近似を求めるとは，折れ線を構成する $k+1$ 個の直線成分の傾きと切片の組 $(a_j, b_j)$, $j = 1, \ldots, k+1$ を決定することである．つまり，この問題は基本的には点列への直線の当てはめ問題だといえる．

以降本論文では，折れ線近似とは次のような場合の点列への折れ線の当てはめを指す．$x\_y$ 平面上に $n$ 個の点 $p_1 = (x_1, y_1), p_2 = (x_2, y_2), \ldots, p_n = (x_n, y_n)$ からなる点集合 $S$ が与えられたとき，点集合 $S$ をより良く近似する折れ線を当てはめる．屈折点の数が $k$ 個の折れ線近似を求める問題を $k$ 連節折れ線近似問題，それを解くアルゴリズムを $k$ 連節場折れ線近似アルゴリズムと呼ぶ．

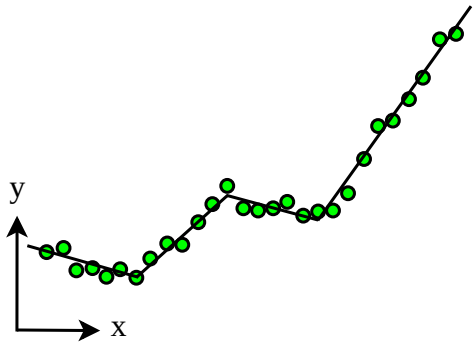



図 1.2: 点列の折れ線近似

### 1.1.3 誤差基準

直線近似問題で，その近似精度の指標として用いられるのが誤差基準である．誤差基準として用いられるのは入力である点集合と，出力である近似直線の位置関係からくる誤差である．この誤差の内，最も一般的に用いられるのが次の式で表される，点集合 $S$ の点 $p_i (i = 1, 2, \ldots, n)$ から近似直線 $y = ax + b$ への垂直距離 $d_i$ である（図 1.3）．

$$d_i = |ax_i + b - y_i|.$$

直線近似問題を上記の誤差基準を用いて解く場合，その値がより小さい近似直線ほど，より良く点集合を近似しているといえる．したがって，最適な直線近似は基準となる誤差

の値を最小にするようにして決定される．このような，最適な直線近似を求めることのできるアルゴリズムを最適アルゴリズムと言う．

誤差基準として，これまでに様々なものが提案されている．それぞれの誤差基準の特徴は，解として求まる近似直線に現れる．つまり，異なる誤差基準を用いれば異なる近似直線が得られると言うのが一般的である．

上で述べた，点と直線の垂直距離を用いる誤差基準として最も一般的なのが，$L_1$ ノルム，$L_2$ ノルム，$L_\infty$ ノルムを用いた三つである．以降では，これら三つの誤差基準の紹介と説明，及びそれを用いた直線近似問題の定式化と，それにより求められる直線の特徴について述べる．

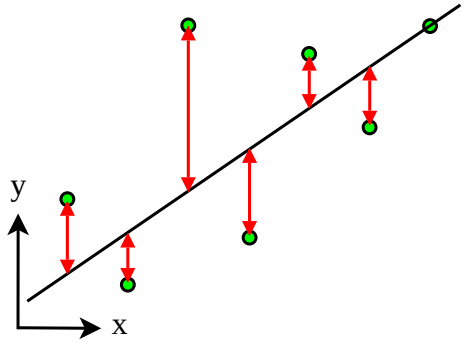



図 1.3: 点集合と直線の垂直距離

### 1.1.4 $L_1$ ノルムに基づく基準

$L_1$ ノルムとは図 1.3 に示されるような，点集合の点 $p_i(i = 1, 2, \ldots, n)$ と近似直線 $y = ax + b$ の垂直距離（$L_1$ 距離）の総和として定義される誤差基準である．次式は，$L_1$ ノルムの定義式である．

$$L_1 \text{ ノルム}: \qquad \sum_{i=1}^{n} |ax_i + b - y_i|. \tag{1.1}$$

2.1 でも述べるが，$L_1$ ノルムを用いた直線近似問題は次のように表される．

$$\text{minimize} \qquad \sum_{i=1}^{n} |ax_i + b - y_i|. \tag{1.2}$$

以降 $L_1$ ノルムを用いた直線近似問題を，$L_1$ 直線近似問題と呼ぶ．

$L_1$ 直線近似アルゴリズムについては 3.1 節で述べるが，他の二つの誤差基準と比較して $L_1$ 直線近似には次のような特徴がある．

- データに含まれるノイズや外れ値の影響を受けにくい．
- 問題の性質により，アルゴリズムが複雑である．

一つ目の特徴については，平面上に与えられた点集合が実験等で得られたデータだった場合，測定時のノイズや外れ値が含まれていることがよくある．一般に解直線がこれらの異常値により大きく動いてしまう直線近似法は好ましくない．$L_1$ 直線近似は，それら異常値の影響を比較的受けにくいという長所を持っていることになる．図 1.4 には，外れ値がある場合の $L_1$ ノルム，$L_2$ ノルムを用いた直線近似の実行結果を示す．実線が $L_1$，破線が $L_2$ による近似直線である．比較すると $L_2$ の方が，外れ値の影響を受けて点列から大きくずれていることがわかる．また，異なる誤差基準を用いると異なる近似直線が得られる事も同様に分かる．

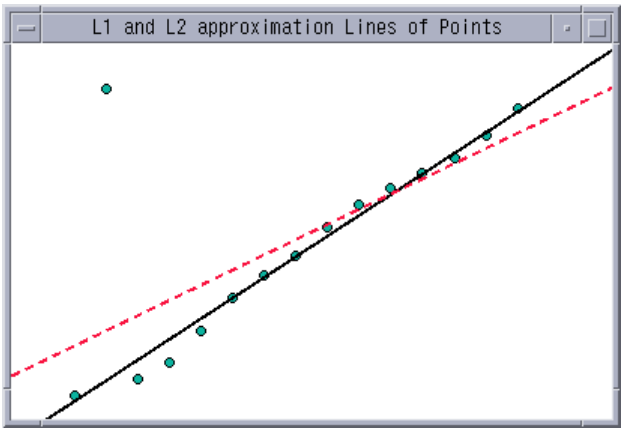



図 1.4: $L_1$ 直線近似と，$L_2$ 直線近似の比較

二つ目の特徴については，3.1 節で詳しく述べるが，線形時間 $L_1$ 直線近似アルゴリズムは他の誤差基準の線形時間直線近似アルゴリズムに比べ，非常に複雑な組み合わせアルゴリズムになっている．その理由を簡単に述べると，目的関数である $L_1$ ノルムが部分的に線形な凸関数なので，その最小値を解析的に求めることができないからである．

本研究では，この $L_1$ ノルムを用いた実用的な一連節折れ線近似アルゴリズムの開発を行う．

### 1.1.5 $L_2$ ノルムに基づく基準

$L_2$ ノルムは，図 1.3 に示されるように，点集合の点 $p_i(i = 1, 2, \ldots, n)$ と近似直線 $y = ax + b$ の垂直距離の二乗（$L_2$ 距離）の総和として定義される．次式は，$L_2$ ノルムに基づ

く誤差の定義式である．

$$L_2 \text{ ノルム:} \qquad \sum_{i=1}^{n}(ax_i + b - y_i)^2. \tag{1.3}$$

$L_2$ ノルムを用いた直線近似問題は次のように表される．

$$\text{minimize} \qquad \sum_{i=1}^{n}(ax_i + b - y_i)^2. \tag{1.4}$$

以降，$L_2$ ノルムを用いた直線近似問題を $L_2$ 直線近似問題と呼ぶ．

$L_2$ ノルムは，直線近似問題の解法として最もよく用いられる最小二乗法の誤差基準である．近似直線 $y = ax + b$ は以下の式により線形時間で簡単に求めることができる．

$$a = \frac{n\sum_{i=1}^{n} x_i y_i - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{n\sum_{i=1}^{n} {x_i}^2 - (\sum_{i=1}^{n} x_i)^2},$$

$$b = \frac{\sum_{i=1}^{n} {x_i}^2 \sum_{i=1}^{n} y_i - \sum_{i=1}^{n} x_i y_i \sum_{i=1}^{n} x_i}{n\sum_{i=1}^{n} {x_i}^2 - (\sum_{i=1}^{n} x_i)^2}.$$

### 1.1.6 $L_\infty$ ノルムに基づく基準

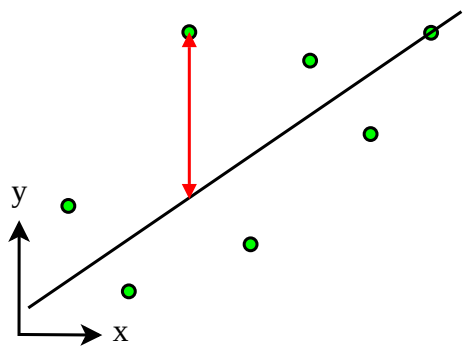



図 1.5: 点集合と直線の $L_\infty$ 距離

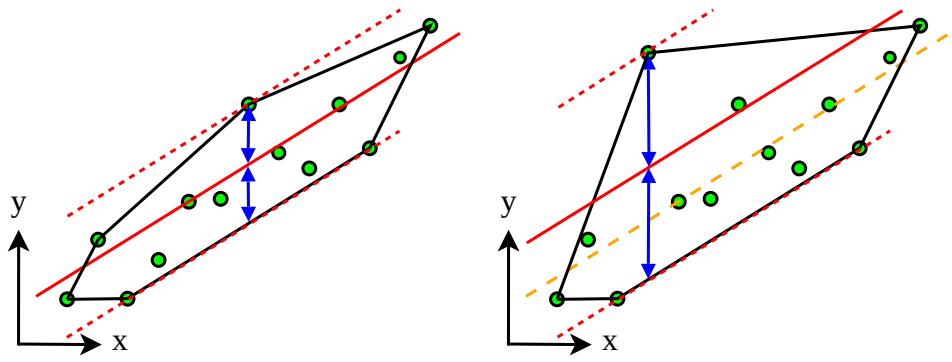
$L_\infty$ ノルムは，図 1.5 に示されるように，点集合の点 $p_i(i = 1, 2, \ldots, n)$ と近似直線 $y = ax + b$ の垂直距離の内，最大のもの（$L_\infty$ 距離）として定義される．次式は，$L_\infty$ ノルムに基づく誤差の定義式である．

$$L_\infty \text{ ノルム:} \qquad \max_{1 \le i \le n} |ax_i - b - y_i|. \tag{1.5}$$

$L_\infty$ ノルムを用いた直線近似問題は次のように表される．

$$\text{minimize} \qquad \max_{1 \le i \le n} |ax_i + b - y_i|. \tag{1.6}$$

以降，$L_\infty$ ノルムを用いた直線近似問題を $L_\infty$ 直線近似問題と呼ぶ．




(a) $L_\infty$ 近似直線　　(b) $L_\infty$ 直線近似における外れ値の影響

図 1.6: $L_\infty$ ノルムを用いた直線近似

$L_\infty$ ノルムを用いた直線近似問題は，図 1.6(a) に示すように，点集合の凸包が分かっていれば，破線で示したような凸包を挟み込むような平行な二本の直線を見つける事により解くことができる．この二本の直線は凸包を挟み込む平行線の内，最も垂直距離が小さいものである．$L_\infty$ 近似直線は，二本の直線から等しい距離にある直線として得られる．凸包を挟み込む平行線を見つける方法はキャリパー法といい [2]，線形時間で実行可能である．したがって，点集合の凸包が既に求まっている場合は線形時間で直線近似が可能である．

図 1.6(b) には，図 1.6(a) の左から四つ目の点を，外れ値として同様に直線近似を行ったものを示した．また，図 1.6(b) 中の大きな破線は図 1.6(a) の近似直線である．両者を比較すると，外れ値により解直線が大きく影響を受けていることが分かる．このように $L_\infty$ 直線近似は，他の誤差基準に比べ非常に，ノイズや外れ値の影響を受けやすいということが言える．

## 1.2　本研究の背景と目的

近年における計算機の発達は情報の電子化を促進し，より多くの仕事を計算機が行う事を可能としている．それに伴い，計算機問題の複雑化と大規模化が進行し，問題をより高速に解くアルゴリズム開発の重要性が増している．実験データ等の整理に用いる回帰分析問題においてもデータの大規模化が進み，これらの問題を効率的に解くアルゴリズムの研

究は以前から行われてきた．その成果の一つとして，平面上に与えられた $n$ 個のデータ点からなる点列の直線近似を，$L_1$ ノルムを用いて線形時間で求めるアルゴリズムが存在する [1]．しかし，その発展問題である $L_1$ ノルムを用いた点列の折れ線近似を求める問題については，現時点において高速かつ実用的なアルゴリズムは存在しない．本研究では，特に折れ線の屈折点の数が一つの場合の一連節折れ線近似を行う高速かつ実用的なアルゴリズムの開発を目的とする．

$L_1$ ノルムとは，直線近似問題の解法に用いられる誤差基準の一つである．誤差基準とは，点列と近似直線の位置関係による誤差で表される近似精度の指標である．$L_1$ ノルムは点列の点と近似直線の垂直距離の総和として表される．誤差基準として最も有名なのが最小二乗法の解法に用いられる $L_2$ ノルムを用いたものである．$L_1$ ノルムと $L_2$ ノルム以外に代表的なものとして，点列の点と近似直線の垂直距離の内，最大のものであらわされる $L_\infty$ ノルムが挙げられる．これら誤差基準の違いは，各誤差基準を用いた直線近似の結果にあらわれる．本研究で用いる，$L_1$ ノルムによる直線近似を他の誤差基準によるものと比較すると，$L_1$ 直線近似は比較的入力データの点列に含まれる外れ値等の異常値の影響を受けにくいことが分かる．つまり，$L_1$ 直線近似には他の誤差基準を用いた直線近似より好ましい結果が得られるという長所がある．しかし，同時に線形時間 $L_1$ 直線近似アルゴリズムは，他の誤差基準を用いたアルゴリズムよりも複雑であるという難点がある．

一般に，点列の折れ線近似アルゴリズムは直線近似アルゴリズムを基にして開発される．したがって直線近似アルゴリズムが複雑である $L_1$ ノルムに関しては，それを用いた折れ線近似のアルゴリズムは，最近まであまり議論されて来なかった．一連節折れ線近似に関しても，入力の点全てを屈折点候補とした全探索に基づく，計算時間 $O(n^2)$ のアルゴリズム以外は存在しなかった．しかし最近になって，折れ線近似アルゴリズム中で使う直線近似アルゴリズムを改善する事により，計算時間 $O(n^{1.33})$ の最適な $L_1$ 一連節折れ線近似アルゴリズムが考案された [4]．しかし，このアルゴリズムは，動的かつ複雑なデータ構造を用いているため実装が困難であり，実用性に欠ける．そこで，本研究ではこの $L_1$ 一連節折れ線近似問題に関して，高速かつ実用性を兼ね備えたアルゴリズムの開発を行う．計算時間 $O(n^{1.33})$ のアルゴリズムは直線近似アルゴリズムを改良し計算時間を改善したのに対し，本研究で提案するアルゴリズムは，屈折点の候補数を絞り込む事により計算時間を減らす．

開発するアルゴリズムの基となるのは，既存の $L_1$ 線形時間直線近似アルゴリズムである．一般に折れ線近似問題の難しさは，その屈折点の数と位置の決定にある．$L_1$ 一連節折れ線近似においてもそれは同様である．屈折点の数の問題に関しては，本研究で考案したアルゴリズムでは，最初に対象の点列が一連節折れ線近似に相応しいかどうかの判定を行い，相応しいと判定されたものに対してのみ折れ線近似を行う．また，この判定には $L_\infty$ ノルムを用いた折れ線近似の結果を用いる．

屈折点の位置の決定に関しては二通りの方法を提案する．一つは点列の判定に用いた点列の $L_\infty$ 折れ線近似の結果を用いる方法である．もう一つは，$L_1$ ノルムによる直線近似の特徴である，近似直線は点列を上下に二分するという性質を用いる方法である．これら

の方法の長所としては，高速である事と単純である事が挙げられる．短所としては，これらの手法により得られる折れ線近似が最適である保証が無いことである．

考案したアルゴリズムの評価を行うために，実際に考案したアルゴリズムを実装し，多数の点列に対して実行した結果を最適 $L_1$ 一連節折れ線近似の結果と比較した．その結果，実用性な速度で妥当な性能をあげるという所期の目的を達成できていることがわかった．

## 1.3 既存の研究

本節では，本研究に関連した既存の研究について，直線近似と折れ線近似のそれぞれについて簡単に紹介する．

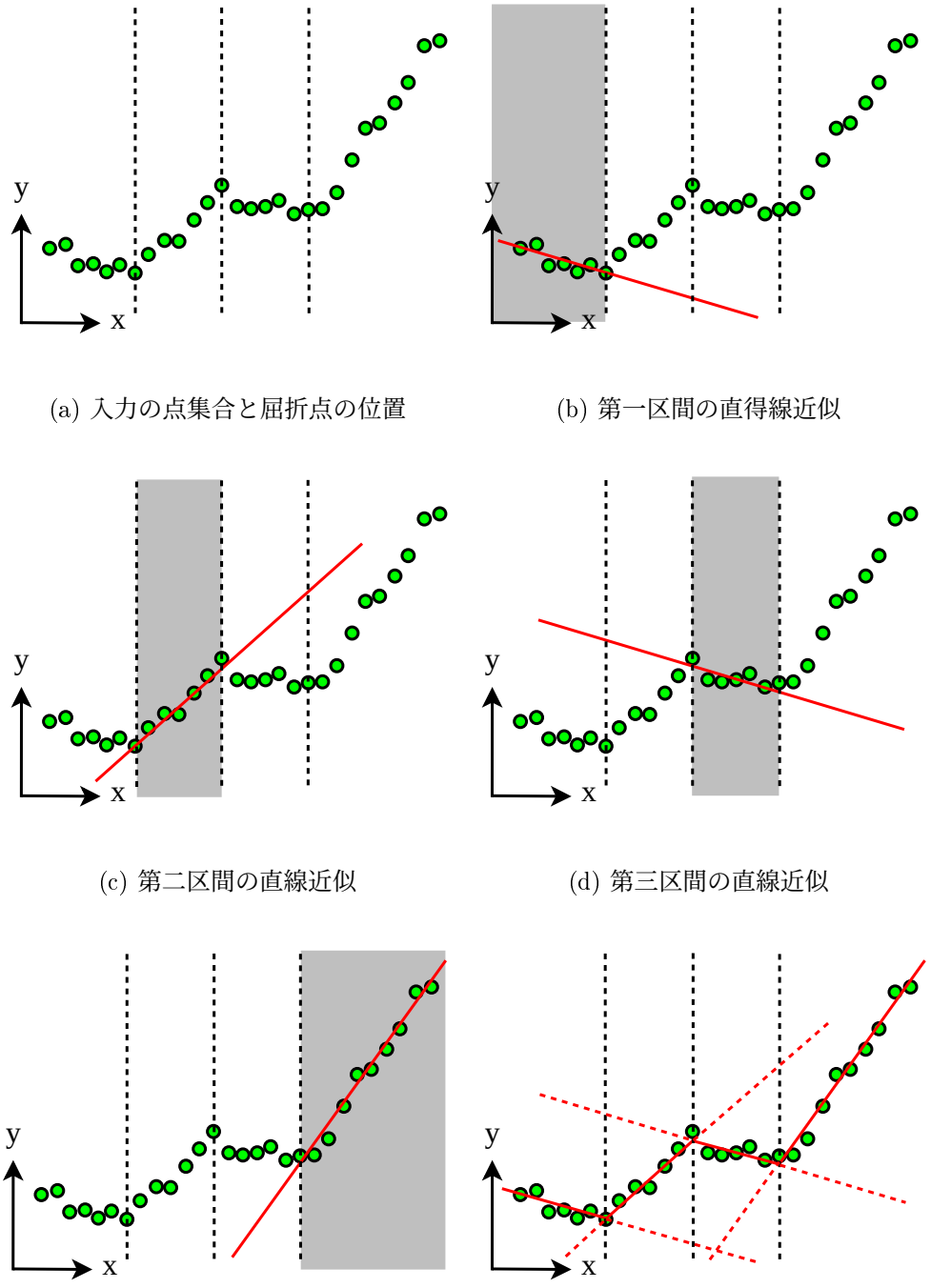
### 1.3.1 直線近似の既存の研究

折れ線とは複数個の直線成分の集まりだといえる．折れ線の屈折点の位置があらかじめ決まっていると仮定した場合について考えると，$x\_y$ 平面上に与えられた点集合の折れ線近似問題の一般的な解法は，次のようになる（図 1.7 参照）．図 1.7(a) の破線は点集合の分割点を表す．はじめに，図 1.7(b)，(c)，(d)，(e) に示すように各区間内の部分点集合に対する近似直線を直線近似アルゴリズムにより求める．次に，1.7(f) に示すように，そうして得られた直線分を隣接するもの同士，それぞれ連結することにより最終的に点集合を近似する折れ線を得る．この考え方に基づく折れ線近似アルゴリズムは，基本的には点列へ直線の当てはめるアルゴリズムと同じである．本研究で提案するアルゴリズムもこの考え方に基づいている．したがって，直線近似アルゴリズムは非常に重要である．

直線近似アルゴリズムに関しては 1.1.3 節で挙げた 3 つの誤差基準それぞれに対して，点集合の点の数に比例する時間で最適な直線近似が得られるアルゴリズムが既に考案されている．本研究で用いる $L_1$ ノルムに関しては，線形時間の $L_1$ 直線近似アルゴリズムが Imai らにより 1989 年に発表されている [1]．この手法では，$L_1$ 直線近似問題が凸計画問題であるという点に着目し，双対変換と枝刈り探索法を用いて線形時間の最適アルゴリズムを実現している．本研究には直接的には関係ないが，他の 2 つの誤差基準についてもみてみると，$L_2$ 直線近似アルゴリズムは最小二乗法として最も有名であり，近似直線も線形時間で簡単に求められる．$L_\infty$ ノルムを用いた点列の直線近似は，点集合に対する凸包が分かれば，その凸包に対してキャリパー法を用いることによりやはり線形時間で可能である [2]．

### 1.3.2 折れ線近似の既存の研究

次に折れ線近似問題の既存の研究を紹介する．点集合を最適に近似する直線は，各誤差基準を最小化するという事は 1.1.3 節で述べた．折れ線近似に関しても同様で，点集合




(a) 入力の点集合と屈折点の位置　(b) 第一区間の直得線近似

(c) 第二区間の直線近似　(d) 第三区間の直線近似

(e) 第四区間の直線近似　(f) 折れ線近似の構成

図 1.7: 直線近似と折れ線近似の関係

の最適な折れ線近似は各誤差基準を最小化する．ここで，まず問題となるのが，折れ線の屈折点の数をどのようにして決定するかである．もし屈折点数に上限を設けなければ，全ての点を通る $n-2$ 個の屈折点を持つ折れ線が誤差基準 0 の最適な折れ線という事になる．しかし，これでは点集合を近似しているとはいえない．したがって，屈折点数 $k$ を $0 \leq k \lneq n-2$ の範囲を持つ整数定数として固定し $k$ 連節の折れ線近似を求めるか，または，点集合に対してある程度の誤差範囲を設けて，誤差がその範囲内に収まり，かつ最小個の屈折点を持つ折れ線近似を求めるというのが一般的である．

本研究で提案するアルゴリズムは，前者のようにはじめに屈折点数を 1 と決めて，点集合の一連節折れ線近似を行う．同様の一連節折れ線近似問題に関しては最近になって $L_1$ ノルムを用いた計算時間が $O(n^{1.33})$ の最適なアルゴリズムが提案された [4]．このアルゴリズムが提案される以前は計算時間が $O(n^2)$ の最も単純な方法である全探索アルゴリズム以外，めぼしいものが無かった．その事を考えると，$O(n^{1.33})$ という計算時間は大変な進歩である．しかし，このアルゴリズムの短所は，非常に複雑で動的なデータ構造とパラメトリックサーチを用いている為に，アルゴリズム自体の実装が難しく，また理論上の計算時間どうりの結果を期待できない点にある．このアルゴリズムはあくまで理論的に優れたアルゴリズムであり，実用的ではないと言える．他に，この一連節折れ線近似問題では $L_2$ ノルムを用いた線形時間の最適なアルゴリズムが存在する [4]．$L_2$ 折れ線近似は解析的に解くことが出来るので，様々なアルゴリズムの組み合わせである $L_1$ 折れ線近似より比較的単純である．

屈折点数を 1 より大きい定数 $k$ にすると，$L_1$，$L_2$ 折れ線近似共に有効なアルゴリズムは現時点では考案されていない．しかし，$L_\infty$ ノルムに関しては計算時間 $O(n \log n)$ の最適な $k$ 連節折れ線近似アルゴリズムが存在する [5]．このアルゴリズムは，屈折点数を $k$ と決めると $k$ 個以下の屈折点を持つ最適な折れ線近似を求める事のできる優れ物であるが，1.1.3 節でも述べたように $L_\infty$ ノルムはデータに含まれるノイズや外れ値等の異常値の影響を受けやすいという短所を持っている．そのため，異常値を含むデータの近似を行うと近似精度は非常に悪くなる．

次に，はじめに折れ線の屈折点数を決めるのではなく，点集合にある程度の誤差範囲を持たせて，その範囲内のみを通り，かつ最小数の屈折点を持つ折れ線近似を求める方法についてみてみる．折れ線が通らなければいけない点集合の誤差範囲とは，簡単に言えば点集合の各点を上下方向に一定値 $w$ ずつ引き延ばして出来る垂直な直線分と，点集合を近似する折れ線は必ず交差しなければいけないということである．このような問題は $L_\infty$ ノルムに上限を与えていると考える事ができる．なぜなら，$L_\infty$ ノルムは点集合と点と近似直線の直線距離の内最大のもので表され，また折れ線は必ず点から一定の距離 $w$ 以内を通るように決められているので，$L_\infty$ ノルムの最大値も $w$ よりは大きくならないからである．したがって，これらの問題は $L_\infty$ ノルムを用いた折れ線近似アルゴリズムということができる．この考え方に基づいた折れ線近似アルゴリズムの一つとして，$x$ 単調な多角形内での可視性判定を用いた線形時間アルゴリズムが考案されている [2][3][6]．本研究で考案したアルゴリズム中では，この線形時間 $L_\infty$ 折れ線近似の結果を何度か使用する．

## 1.4 本論文の構成

ここでは以降の本論文の構成を紹介する．第２章では，本研究の対象である，$L_1$ 一連節折れ線近似問題の定式化と，その一般的な解法について述べる．第３章では，本研究の提案手法の基になった，$L_1$ 直線近似アルゴリズムと $L_\infty$ 折れ線近似アルゴリズムの二つを簡単に紹介をする．第４章では，第３章で紹介した二つのアルゴリズムを基に開発した二通りの本研究の提案手法を紹介する.，第５章では，第４章で紹介した２つアルゴリズムを実装し，点列に対し実行した結果の評価を行う．第５章では，本研究のまとめと，今後の課題について述べる．

# 第2章　$L_1$一連節折れ線近似問題

本章では，一般的な一連節折れ線近似問題の解法を通して，$L_1$ 直線近似を用いた $L_1$ 一連節折れ線近似問題の解法手順の説明と定式化を行う．

## 2.1　$L_1$ 直線近似問題とその定式化

$L_1$ 一連節折れ線近似は，$L_1$ 直線近似アルゴリズムに基づいている．そこで，まず $L_1$ 直線近似問題を定式化する．

入力は平面上の点集合 $S = \{p_1, p_2, \ldots, p_n\}$，$p_i = (x_i, y_i)$ であり，出力は式 1.1 の $L_1$ ノルムを最小にする直線 $y = ax + b$ である．ただし，全ての点は $x$ 座標の昇順に既にソートされているものと仮定する．したがって，この問は次のように表すことができることは 1.1.4 節でも述べた．

$$\text{minimize} \qquad \sum_{i=1}^{n} |ax_i + b - y_i|.$$

これは，また以下の様な線形計画問題に定式化できる．

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{n} w_i, \\ \text{subject to} \quad & -w_i \leq ax_i + b - y_i \leq w_i, \quad i = 1, \ldots, n. \end{aligned}$$

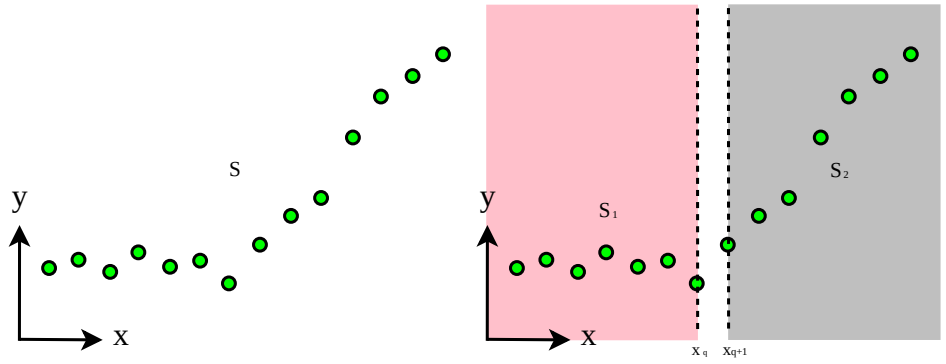
ここで $L_1$ ノルムは凸関数なので最適解は一意に求まる．$L_1$ 直線近似アルゴリズムについては 3.1 節で述べる．

## 2.2　一連節折れ線近似問題とその解法手順

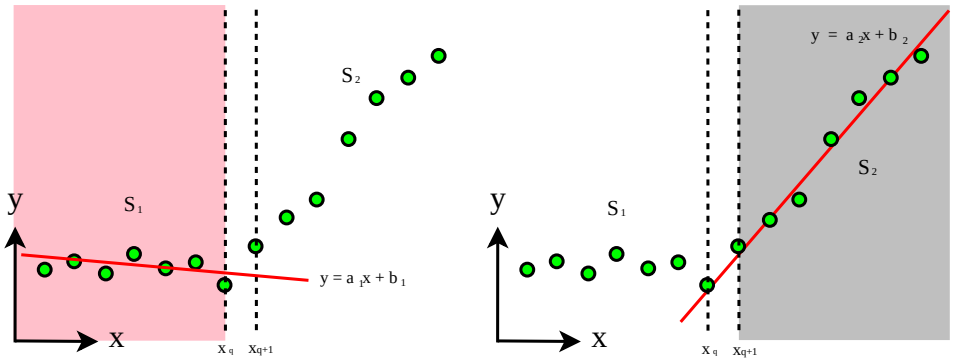
ここでは，一般的な一連接折れ線近似問題とその解法について述べる．

一連接折れ線近似問題の入力は平面上の点集合 $S = \{p_1, p_2, \ldots, p_n\}$，$p_i = (x_i, y_i)$ である．そして出力は，誤差基準を最小にする，$x$ 単調かつ連続な，屈折点が一つの折れ線 $Q$ である．

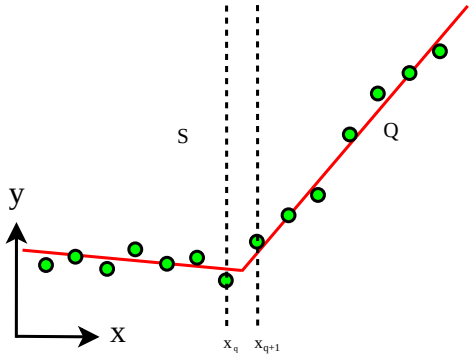
以下では，図 2.1 に沿って一連接折れ線近似問題を解く手順を説明する．図 2.1(a) のように，入力として平面上に点集合 $S$ が与えられたとする．点集合 $S$ においては，$x_i \leq x_{i+1}$




(a) 入力点集合 $S$　　(b) 点集合 $S_1$ と $S_2$




(c) 点集合 $S_1$ に対する直線近似　　(d) 点集合 $S_2$ に対する直線近似




(e) 点集合 $S$ の一連節折れ線近似

図 2.1: 一連節折れ線近似問題の解法

が $i = 1, 2, \ldots, n-1$ で常に成り立つとする．はじめに，図 2.1(b) に示すように，$x = x_q$ において点集合 $S$ を $S_1 = \{p_1, p_2, \ldots, p_q\}$ と，$S_2 = \{p_{q+1}, p_{q+2}, \ldots, p_n\}$ のような左右二つの部分点集合に分割する．次に，図 2.1(c)，(d) に示すように，分割した点集合 $S_1$，$S_2$ それぞれに対しての最適な直線近似 $y = a_1x + b_1$，$y = a_2x + b_2$ を直線近似アルゴリズムにより求める．最後に，図 2.1(e) に示すように，これら二本の直線の交点を連節点とすることにより，屈折点が一つの最適な折れ線を求めることができる．以上の様な操作を，分割点の位置を $q = 2, 3, \ldots, n-1$ の様に順次変更しながら繰り返すと，それぞれの分割に対しての最適な折れ線が求まる．そうして求まった $n-2$ 本分の折れ線を比較して，最もノルムが小さいものを求め，それを入力の点集合 $S$ に対する最適な折れ線 $Q$ として出力する．以上が，一連接折れ線近似問題を解くための一般的な手順である．

## 2.3 折れ線近似の成立条件

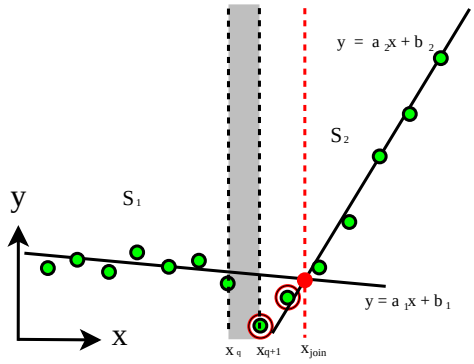



図 2.2: 不都合な折れ線近似

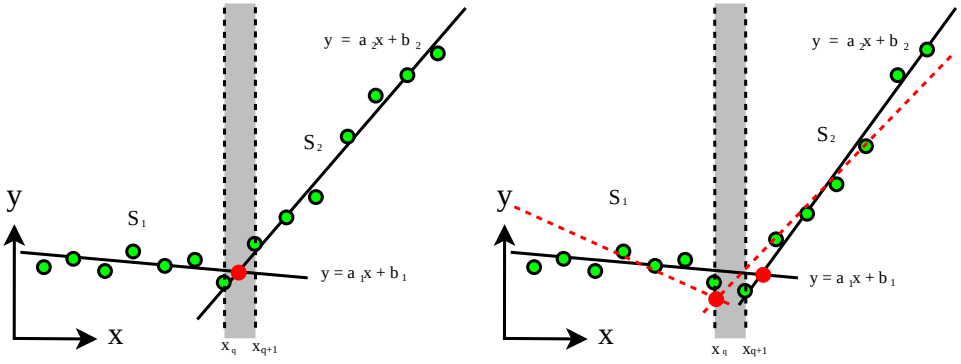
2.2 節より，左右二つの点集合 $S_1$，$S_2$ それぞれの直線近似 $y = a_1x + b_1$，$y = a_2x + b_2$ より折れ線近似が求められることがわかる．しかし，単純に二本の直線より折れ線を構成すると不都合が生じる場合がある．それは，図 2.2 に示す様に，折れ線の屈折点が分割の範囲 $x_q \leq x \leq x_{q+1}$ の外にある場合である．折れ線の屈折点の $x$ 成分 $x_{join}$ は次の式で与えられる．

$$x_{join} = \frac{b_1 - b_2}{a_2 - a_1}. \tag{2.1}$$

この折れ線近似の不都合な点とは，左側点集合 $S_1$ の直線近似 $y = a_1x + b_1$ が，右側点集合 $S_2$ の範囲まで近似してしまっている事にある．つまり，点集合 $S_2$ の点の内，$x_{q+1} \leq x \leq x_{join}$ の範囲にあるものは，近似に影響を与えていないにもかかわらず，直線 $y = a_1x + b_1$ に近似され，影響を与えた直線 $y = a_2x + b_2$ には近似されないという矛盾が生じている．図 2.2 では，強調されている点が影響を無視された点である．したがって，左右二本の直

線から折れ線を構成する際には次式のような折れ線の成立条件が必要となる．

$$x_q \leq \frac{b_1 - b_2}{a_2 - a_1} \leq x_{q+1}. \tag{2.2}$$




(a) 成立条件を満たす場合　　(b) 成立条件を満たさない場合

図 2.3: 折れ線近似の成立条件

図 2.3 には，成立条件を満たすような左右二本の直線近似を示す．次に，成立条件を満たさなかった場合について考える．成立条件を満たさなかった場合は，図 2.3(b) に示すように折れ線の屈折点が必ず点集合の分割点である $x = x_q$ の上にくるよう固定し，その制約のもとで再度，左右直線近似を求めることにより折れ線近似を構成する．つまり，屈折点の $x$ 座標 $x_{join}$ が $x_{join} = x_q$ を満たすような折れ線近似を求めることになる．図 2.3(b) では破線で示された直線が制約条件のもと再度もとめた直線近似である．この制約条件は式 2.1 より次式のように表される．

$$\frac{b_1 - b_2}{a_2 - a_1} = x_q. \tag{2.3}$$

## 2.4 $L_1$ 一連節折れ線近似問題とその定式化

2.2 節では，一般的な一連節折れ線近似問題の解法とその手順について述べた．その，解法に $L_1$ ノルムを用いた場合の一連節折れ線近似問題の定式化を行う．求める折れ線の $L_1$ ノルム最小の条件より，$L_1$ 一連節折れ線近似問題は次のように表すことができる．

$$\text{minimize} \quad \sum_{i=1}^{q} |a_1 x_i + b_1 - y_i| + \sum_{i=q+1}^{n} |a_2 x_i + b_2 - y_i|. \tag{2.4}$$

また、$a_1 \neq a_2$ ならば，次のような凸計画問題に定式化できる．

$$\text{minimize} \quad \sum_{i=1}^{n} w_i, \tag{2.5}$$

$$\text{subjecto to} \quad -w_i \leq a_1 x_i + b_1 - y_i \leq w_i \quad \text{for } i \leq q, \tag{2.6}$$

$$-w_i \leq a_2 x_i + b_2 - y_i \leq w_i \quad \text{for } i \geq q+1, \tag{2.7}$$

$$\text{and} \quad x_q \leq \frac{b_2 - b_1}{a_2 - a_1} \leq x_{q+1} \text{ (成立条件)}. \tag{2.8}$$

$L_1$ 一連節折れ線近似問題とは，$L_1$ ノルムを最小化し，かつ式 2.8 を満たすような左右二本の $L_1$ 直線近似を求める問題である．つまり，この問題は，以上のような条件を満たす $(a_1, b_1, a_2, b_2)$ の組を見つけることを目的としている．

## 2.5 最適 $L_1$ 一連節折れ線近似アルゴリズム

2.1 節で紹介した一連節折れ線近似問題の解法手順に，2.3 節で紹介した折れ線の成立条件を加えた $L_1$ 一連節折れ線近似アルゴリズムを以下に示す．

for: $q = 2, 3, \ldots, n-1$(分割点 $p_q$ を順次変更)

1. 点集合 $S$ を左側点集合 $S_1$ と，右側点集合 $S_2$ に分割する．
2. 点集合 $S_1$，$S_2$ それぞれに対し $L_1$ 直線近似を用いて最適な近似直線を求める．
3. 求めた二本の直線からなる折れ線が式 2.8 の成立条件を満たしていればそれを最適な折れ線近似として出力する．
4. それ以外の場合は式 2.3 を制約条件として加えた上で，最適な折れ線近似を再び求める．

上に示したアルゴリズムは，与えられた点集合の先頭と最後の点を除いた全ての点を最適分割点の候補とする全探索で，最適な折れ線近似を見付ける事を可能としている．計算時間は $n-2$ 個の分割に対し $O(n)$ の直線近似をそれぞれ二回ずつ行うので $O(n^2)$ となる．4 章で紹介する本研究で提案する手法では，分割点候補を定数個に絞り込み，それら分割点候補に対して上のアルゴリズムの手順 1，2，3，4 を適用することにより計算時間 $O(n)$ のアルゴリズムを得た．このアルゴリズムでは最適な折れ線近似が求まる保証はないが，より最適に近い折れ線近似を求められるようにアルゴリズムを設計している．

# 第3章　関連研究

提案手法を紹介する前に，本研究で提案する線形時間の $L_1$ 一連節折れ線近似アルゴリズムの基となる二つのアルゴリズムを紹介する．一つは $L_1$ ノルムを用いた線形時間直線近似アルゴリズム [1]，もう一つ $L_\infty$ ノルムを用いた線形時間折れ線近似アルゴリズムである [3]．ここでは，それらのアルゴリズムの概略のみの紹介とする．詳細についてはそれぞれの文献を参照されたい．

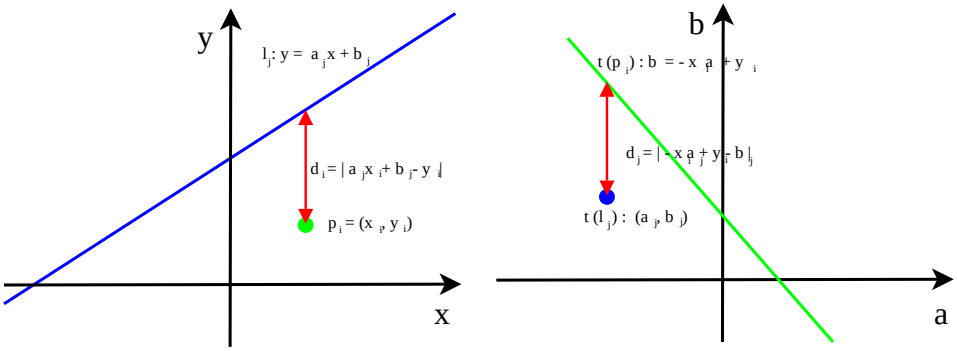
## 3.1　$L_1$ ノルムを用いた直線近似アルゴリズム

ここで紹介する $L_1$ 直線近似アルゴリズムは，1989 年に Imai らにより提案された線形時間の最適アルゴリズムである [1]．提案された当時においては，$L_2$，$L_\infty$ ノルムを用いた最適直線近似は既に線形時間で可能であったが，$L_1$ ノルムに関しては有効な手法は確立されておらず，最も単純な手法として計算時間 $O(n^3)$ のアルゴリズムが存在するのみであった．Imai らは，これを双対変換と枝刈り探索を用いることにより線形時間で実行可能とした．以下では双対変換と枝刈り探索の考え方を説明し，$L_1$ ノルムが部分的に線形な連続した凸関数であることから，それらを用いて線形時間 $L_1$ 直線近似が可能である事を概略で紹介する．なお，これまでと同様に入力は点集合 $S$，出力は直線 $y = ax + b$ とする．

### 3.1.1　双対変換

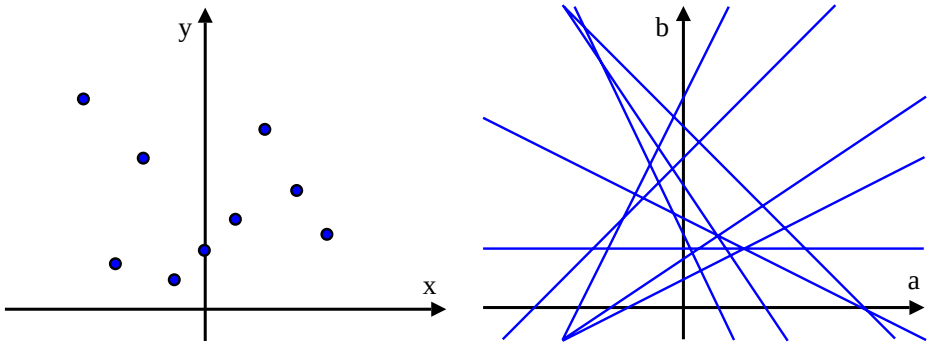
図 3.1 に示すように，$x\_y$ 平面上の点 $p_i : (x_i, y_i)$ を，$a\_b$ 平面上の直線 $t(p_i) : b = -x_i a + y_i$ に，$x\_y$ 平面上の直線 $l_j : y = a_j x + b_j$ を $a\_b$ 平面上の点 $t(l_j) : (a_j, b_j)$ に置き換えるような変換を双対変換という．双対変換の特徴は，$x\_y$ 平面における点 $p_i$ と直線 $l_i$ の間の垂直距離 $d_i = |a_j x_i + b_j - y_i|$ が，変換後の直線 $t(p_i)$ と点 $t(l_i)$ の垂直距離 $d_j = |-x_i a_j + y_i - b_j|$ として保存されることである．

図 3.2(a) に示すような点集合に対して双対変換を用いると，図 3.2(b) に示すような $a\_b$ 平面上の $n$ 本の直線からなるアレンジメントを構成する．そうしてできたアレンジメントの $n^2$ 個の頂点は，図 3.6 に示すようにそれぞれ $x\_y$ 平面における，入力の点集合の内，二点を通る一本の直線を表している．



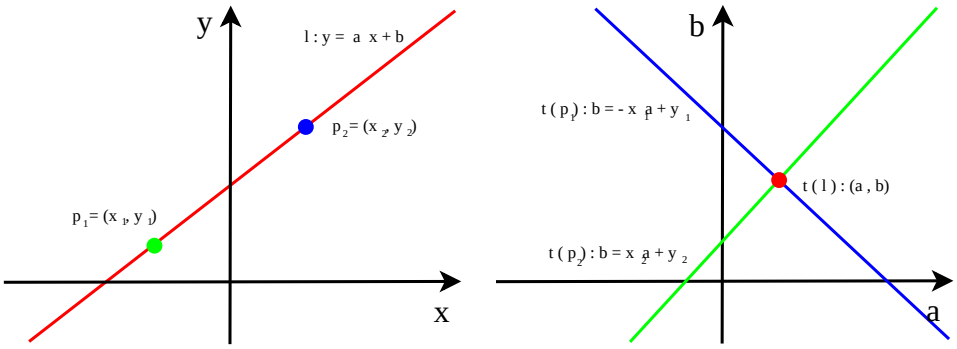

(a) $x\_y$ 平面　　(b) $a\_b$ 平面

図 3.1: 双対変換




(a) $x\_y$ 平面上の点集合　　(b) $a\_b$ 平面上のアレンジメント

図 3.2: アレンジメント




(a) $x\_y$ 平面上の直線　　(b) $a\_b$ 平面上の交点

図 3.3: アレンジメントの交点

### 3.1.2 凸関数 $D(a,b)$

点集合 $S$ に対する近似直線 $y=ax+b$ の $L_1$ ノルムを次のように $D(a,b)$ で表す.

$$D(a,b)=\sum_{i=1}^{n}|ax_i+b-y_i|. \tag{3.1}$$

したがって，$L_1$ 直線近似問題は次のように関数 $D(a,b)$ の最小値を求める問題として表せる.

$$\text{minimize} \qquad D(a,b). \tag{3.2}$$

また，以下の証明により，$D(a,b)$ が凸関数であることが分かる.

$$\begin{aligned}
&\theta D(a_1,b_1)+(1-\theta)D(a_2,b_2)-D(a_1+(1-\theta)a_2,\theta b_1+(1-\theta)b_2)\\
&\quad=\sum_{i=1}^{n}\{\theta|y_i-(a_1x_i+b_1)|+(1-\theta)|y_i-(a_2x_i+b_2)|\\
&\qquad\qquad-|y_i-[(\theta a_1+(1-\theta)a_2)x_i+(\theta b_1+(1-\theta)b_2)]|\}\\
&\quad\geq 0.
\end{aligned}$$

(任意の $A$ と $B$ に対して $|A|+|B|\geq|A+B|$ が成り立つ．)

$a\_b\_c$ 空間での関数 $D(a,b)$ について考えると，図 3.4 に示すような凸多面体になっており，$a\_b$ 平面上のアレンジメントの各頂点は，$D(a,b)$ の凸多面体の各頂点に対応している．し

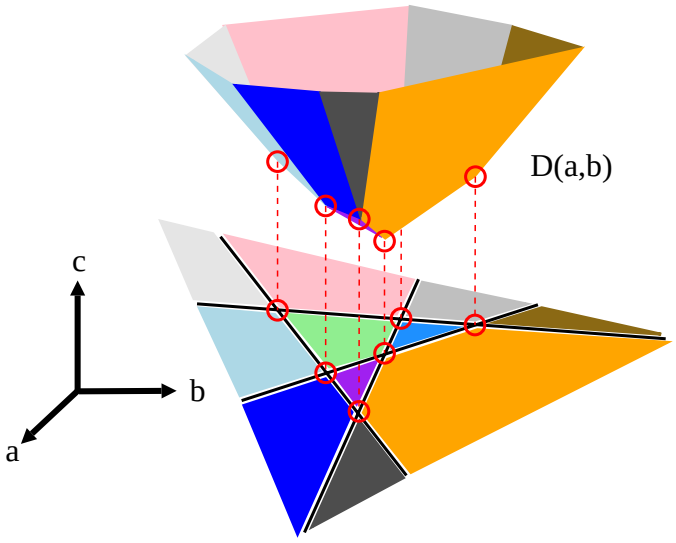



図 3.4: 関数 $D(a,b)$

たがって，図 3.5 に示すように関数 $D(a,b)$ の最小値も，$a\_b$ 平面上のアレンジメントの頂

点の一つにあることがわかる．これは同時に，点集合 $S$ の最適 $L_1$ 直線近似は点集合 $S$ の 2 点を通ることを表している．また，このことから最適な直線近似を求めるにはアレンジメントを構成する直線の内，この頂点を構成する二本の直線のみが必要であり，他の直線は必要ない事が分かる．

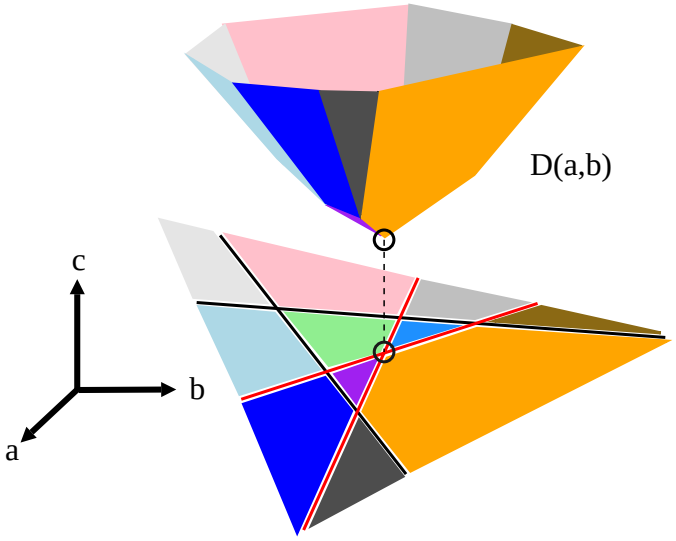
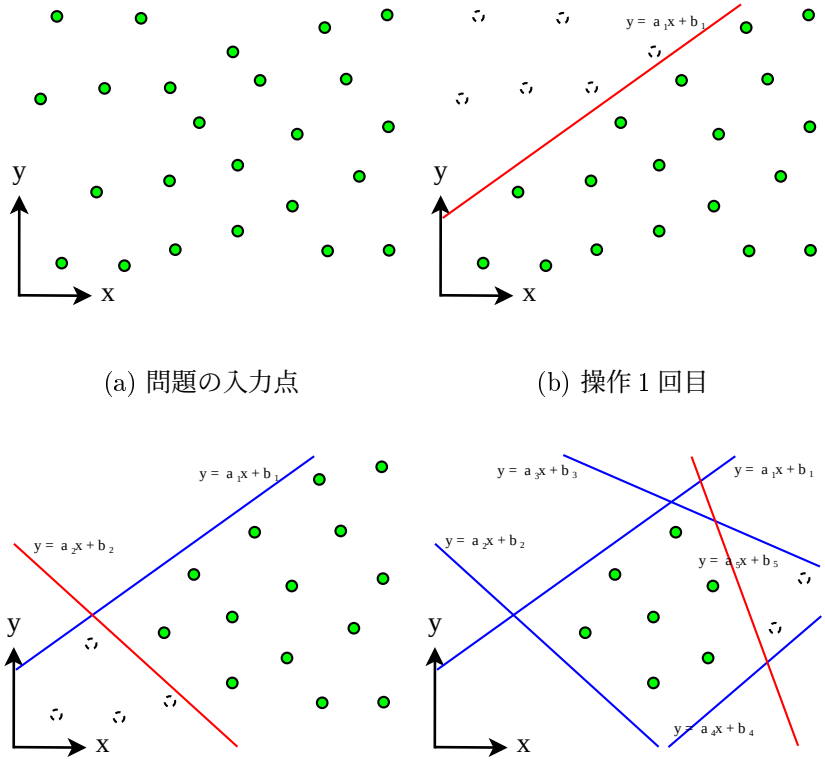



図 3.5: 関数 $D(a, b)$

### 3.1.3 枝刈り探索

$L_1$ 直線近似問題のように最適解が問題の要素の内，定数個の要素により決まるような場合には枝刈り探索法が有効である．枝刈り探索法とは，最適解の決定に不必要な要素を取り除いていくことにより，必要な要素のみを絞り込んでいく方法である．要素の取り除きは，不必要な要素を判定することが可能な検査を，その時点での残りの要素に用いることにより行うことができる．そのような検査を再帰的に用いて，毎回の操作で残りの要素の一定割合分ずつを取り除く．もし，毎回の操作が問題のサイズに対する線形時間でできるなら，最終的にその問題は線形時間で解くことが可能である．以下に枝刈り探索の動作を表す例を示す．

図 3.6(a) に示すように，平面上に問題の点集合が与えられたとき，これらの点の内，定数個のみが最適解の決定に必要であるとする．ある種の検査により，直線 $y = a_1x + b_1$ 以上にある要素は最適解の決定に関係しないと判定されたならば，図 3.6(b) に示すようにそれら不必要な要素は取り除いてもよい．また，同様の検査により直線 $y = a_2x + b_2$ 以下の要素も最適解の決定に関与しないことが判明したとすると，図 3.6(c) に示すようにこれらの要素も同様に取り除いてよい．以降，図 3.6(d) に示すように，毎回検査を繰り返し，最適解を決定する要素を絞り込んでいく．このような動作の手法を枝刈り探索法という．




(a) 問題の入力点　(b) 操作 1 回目

(c) 操作 2 回目　(d) 操作 5 回目

図 3.6: 枝刈り探索の動作例

### 3.1.4 $L_1$ 直線近似アルゴリズムの概要

3.1.2 節で述べたように，関数 $D(a,b)$ が凸関数であることから $x\_y$ 平面上に与えられた点集合 $S$ の $L_1$ 直線近似は，点集合 $S$ の点の内 2 点を通る直線として求められる．これを，$a\_b$ 平面上でみると関数 $D(a,b)$ 値を最小にする頂点を構成する二本の直線のみが最適解の決定に必要であることが分かる．したがって，枝刈り探索法を用いてこれら必要な直線以外の直線を，$a\_b$ 平面上で検査，判定し取り除いていくことにより，最終的に最低な $L_1$ 直線近似を得ることができる．この，枝刈り探索の毎回の冗長な要素の取り除きにかかる時間が，入力の点集合の数 $n$ に対する線形時間で実行可能である．したがって，点集合 $S$ の最適な $L_1$ 直線近似は線形時間で求める事が可能である．

図 3.7 には点列の $L_1$ 直線近似の実行例を示す．これを見ればわかるように，点列の $L_1$ 直線近似は点列の 2 点を通り，かつ点列を上下に二等分する性質をもつ．

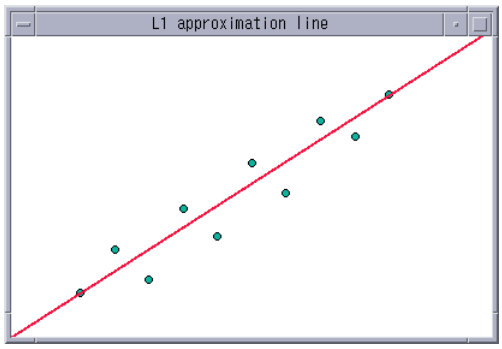



図 3.7: 点列の $L_1$ 直線近似の性質

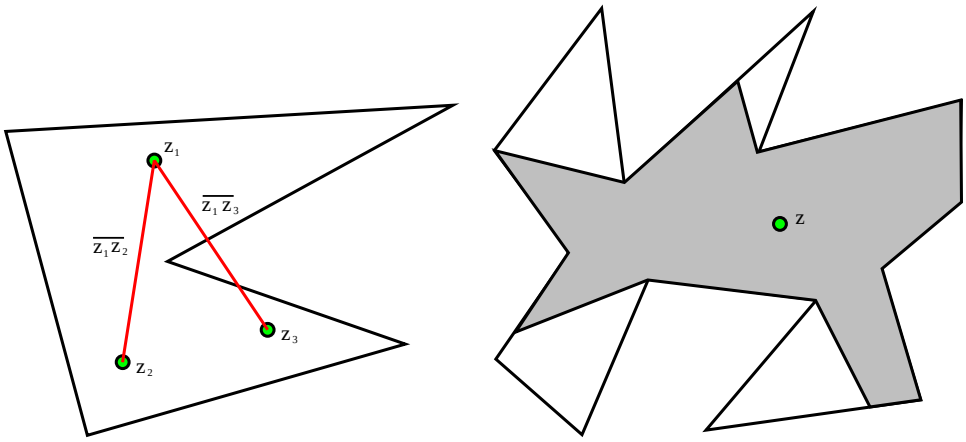
## 3.2 $L_\infty$ ノルムを用いた折れ線近似アルゴリズム

本研究の提案手法で用いる既存のアルゴリズムとして，もう一つの重要なものが文献 [3] に紹介されている $L_\infty$ ノルムを用いた折れ線近似アルゴリズムである．その手法では，点列が一連節折れ線近似を行うのに相応しいかの判定と，折れ線近似の屈折点の候補の絞り込みでこのアルゴリズムを用いている．

本アルゴリズムは，入力として $n$ 個の点 $p_i = (x_i, y_i)$，$i = 1, 2, \ldots, n$ からなる点集合 $S$ が与えられたとき，点集合 $S$ に対する誤差範囲一定の折れ線近似を $O(n)$ の計算時間で求めることができる．誤差範囲の上限が $L_\infty$ ノルムの上限と一致することから，これは $L_\infty$ ノルムを用いた折れ線近似アルゴリズムということができる．

本節ではこのアルゴリズムの概要を紹介する．詳細については，[3] を参照されたい．以下の 3.2.1 節では，線形時間 $L_\infty$ 折れ線近似を実現させる為に必要な多角形内の可視性について紹介する．その可視性を用いて，3.2.2 節ではアルゴリズム動作を紹介する．

### 3.2.1 多角形内での可視性




(a) 多角形内の点同士の可視性　　(b) 多角形内の点からの可視領域

図 3.8: 多角形内での可視性

ここでは，多角形内での可視性について述べる．多角形内の 2 点 $z_1$，$z_2$ が互いに可視であるとは，図 3.8(a) に示すように線分 $\overline{z_1 z_2}$ が多角形の辺と交わらないことをいう．したがって同図の点 $z_1$，$z_3$ は互いに可視ではない．

図 3.8(b) には，多角形内における点 $z$ からの可視領域を示す．多角形内での，一つの辺から可視な領域を求める問題では，$O(n \log n)$ の計算時時間がかかる．

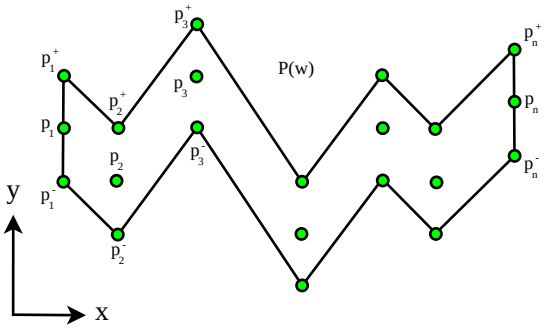
### 3.2.2 $L_\infty$ 折れ線近似アルゴリズムの動作

入力として $n$ 個の点 $p_i = (x_i, y_i)$，$i = 1, 2, \ldots, n$ からなる点集合 $S$ が与えられたとする．この時，点集合の各点を上下にそれぞれ，ある一定値 $w$ だけ平行移動することにより点 $p_i^+ = (x_i, y_i + w)$，$p_i^- = (x_i, y_i - w)$，$i = 1, 2, \ldots, n$ からなる点集合 $S^+$ と $S^-$ が得られる．また，それらの点を直線分で結ぶことにより図 3.9(a) に示すような多角形が得られる．これを多角形 $P(w)$ とする．

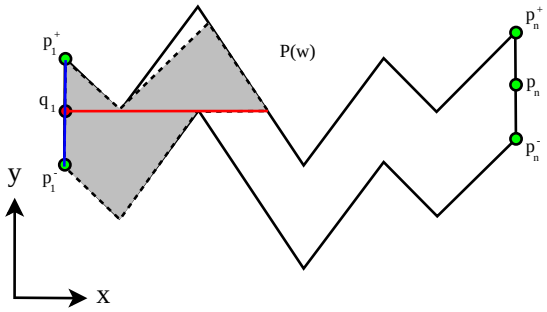
このような誤差範囲一定の多角形内で，辺 $\overline{p_1^+ p_1^-}$ と辺 $\overline{p_n^+ p_n^-}$ を結ぶような線分数が最小の，$x$ 単調な折れ線が，誤差範囲一定のもとでの点列の最適な折れ線近似となる．

そのような折れ線近似は図 3.9 に示すように，多角形 $P(w)$ 内での辺からの可視性判定を繰り返す事により求められる．多角形内での辺からの可視領域を求めるには，$O(n \log n)$ の計算時間が必要になることは 3.2.1 節で述べたが，この手法では，多角形の単調性を十分利用することにより全体で $O(n)$ の計算時間で，最適な折れ線近似が求められる．

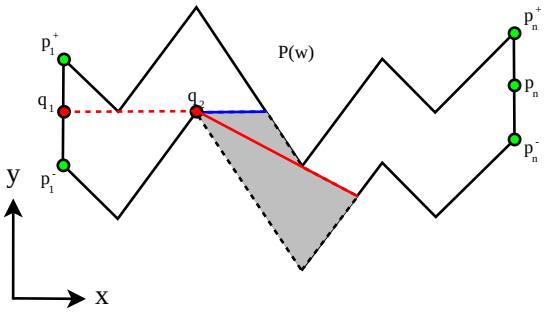
アルゴリズムの動作を簡単に説明すると，はじめに図 3.9(b) に示すように，辺 $\overline{p_1^+ p_1^-}$ からの多角形 $P(w)$ 内での可視領域を求める．こうして求まった可視領域もやはり多角形で



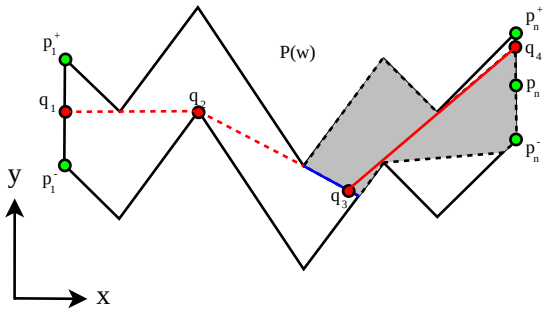

(a) 多角形 $P(w)$



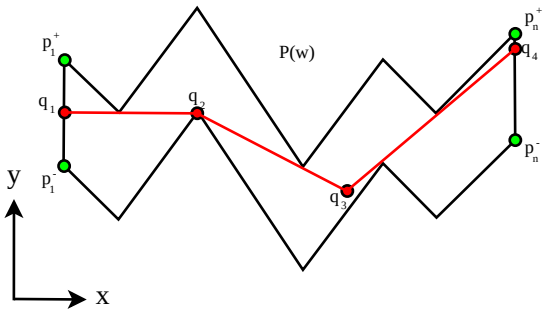

(b) 可視性判定 1




(c) 可視性判定 2




(d) 可視性判定 3




(e) 求まった折れ線近似

図 3.9: アルゴリズムの動作

ある．また，これにより多角形 $P(w)$ は，辺 $\overline{p_1^+ p_1^-}$ からの可視領域と，可視でない幾つかの領域に分けられる．その可視でない領域の内，辺 $\overline{p_n^+ p_n^-}$ を含む領域を，$P'$ とする．また，可視領域の辺で，この $P'$ と隣接するものをウィンドウ辺と呼ぶ．このウィンドウ辺の延長線と辺 $\overline{p_1^+ p_1^-}$ の交点を折れ線の出発点 $q_1$ とする．以降，図 3.9(c)，(d) では，もとまったウィンドウ辺からの多角形 $P'$ 内での可視領域を求め，同様の操作をくり返し最終的に，図 3.9(e) に示すような誤差範囲一定の最小線分数の折れ線近似が得られる．

# 第4章　提案手法

本章では，本研究で開発した $L_1$ 一連節折れ線近似アルゴリズムの紹介を行う．本研究では従来の手法に比べ実用性を重視したアルゴリズム開発を行った為，特に一連節折れ線近似に相応しい点列に対してのみ，有効な折れ線近似が得られる設計となっている．したがって，近似対象として相応しくない点集合は，はじめの段階で省くことを考える．その為の判定の基準とその方法を，入力の点列に対する前提条件として 4.1 節で紹介する．本提案手法を従来の手法と比較した特徴として，折れ線の屈折点の位置をあらかじめ予測するという点が挙げられる．その屈折点の予測方法を二通り，屈折点候補の絞り込み方法として 4.2 節で紹介する．そうして求まった，定数個の屈折点候補を点集合の分割点とし，分割された左右の点集合に対し $L_1$ 直線近似アルゴリズムを用いて，直線近似を行う事により左右の二本の直線を得る．それら二本の直線より折れ線近似を求める方法を折れ線近似の構成法として 4.3 節で紹介する．最後に提案手法をアルゴリズムとして 4.4 節でまとめる．

## 4.1　前提条件

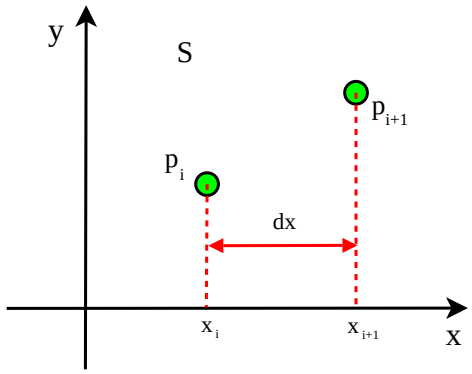
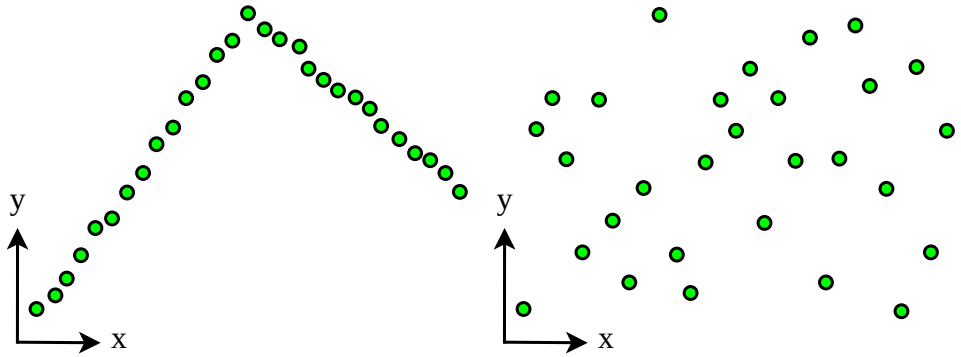



図 4.1: 入力の点列の特徴

本研究では近似対象の点列として特に，実験等で得られるデータの点列を想定している．そのため図 4.1 に示すように，入力である点集合 $S$ においては，$x_i \lneq x_{i+1}$ が $i = 1, 2, \ldots, n-1$ で常に成り立ち，かつデータ点の間隔 $dx = x_{i+1} - x_i$ は，常に一定であると仮定できる．




(a) 提案手法に適した点列　　(b) 提案手法に適さない点列

図 4.2: 一連節折れ線近似が適した点列

また実用性の観点から，本研究で開発したアルゴリズムは，図 4.2(a) に示すような，特に屈折点が一つの折れ線で近似することが相応しい点列に対してのみ用いることを前提にしている．したがって図 4.2(b) のような点集合を省くための条件が必要になる．ここでは，その条件を一連節折れ線近似性条件と呼び，条件を満たす場合のみ折れ線近似を行う事とする．

その条件判定に用いるのが，3.2 節で紹介した線形時間の $L_\infty$ 折れ線近似である．具体的には，$L_\infty$ 折れ線近似で得られた折れ線の屈折点数が一つだった場合のみ，その点集合は一連節折れ線近似に相応しい折れ線であると判断する．

ただし，ここで問題になるのが $L_\infty$ 折れ線近似は外れ値等のデータ中に含まれる異常値に非常に弱いという点である．$L_\infty$ 折れ線近似ではデータ中に一つでも異常値が含まれていた場合，大体においてその点は一つの屈折点となり，その左右を含め計 3 個の屈折点の原因になる．結果として，異常値を除いた点列が一連接折れ線近似に相応しい形状であったとしても，屈折点数が 1 以上となり，一連節折れ線近似性条件を満たさなくなり，折れ線近似が行われない．

その点を解決する為に，異常値の点と，その周囲の点との比較により異常値を取り除く方法を考案した．4.1.1 節では，考案した点列に含まれる外れ値を判定し除去する方法を紹介する．あくまで，異常値除去は，$L_\infty$ 折れ線近似の欠点を補う為のものであるので，$L_1$ 折れ線近似への影響はない．$L_1$ 折れ線近似は異常値を除かない入力点集合に対して行われる．

4.1.2 節では，$L_\infty$ 折れ線近似を用いた一連節折れ線近似性条件とその判定方法を，より詳細に説明する．

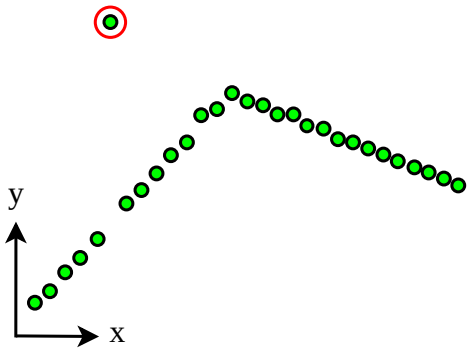



図 4.3: 異常値の例

### 4.1.1 点列に含まれる異常値の除去

異常値とは，図 4.3 で強調されている点のように周囲の点と比較して明らかに異なる挙動をするデータの $y$ 値のことをいう．したがって，本手法では周囲の点の $y$ 値と異常値の $y$ 値が明らかに違うという点に注目し，入力点が異常値をもつかどうかの判定を行う．

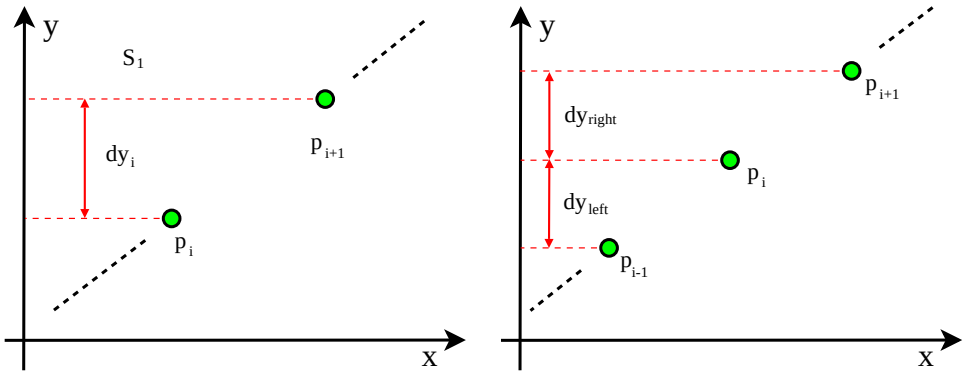
一連接折れ線近似に相応しい点集合で，かつ異常値を含むような点集合 $S$ が与えられたとする．$S$ を折れ線の屈折点で左右に分割したとき，その左側点集合を $S_1 = \{p_1, p_2, \ldots, p_q\}$，右側点集合を $S_2 = \{p_{q+1}, p_{q+2}, \ldots, p_n\}$ とする．この時，点集合 $S_1$ のうち，異常値を持つ点を除いた点においては，図 4.4(a) に示すような，隣接するデータ間の $y$ 値の差の絶対値 $dy_i = |y_{i+1} - y_i| (i = 1, 2, \ldots, q-1)$ は，ほぼ一定の値になることが予測される．これは，点集合 $S_2$ においても同様である．このことを利用して，異常値の判定を行う．

現在判定を行っている点を，$p_i = (x_i, y_i)$ とする．その点の左側で隣接する点 $p_{i-1}$ との図 4.4(b) に示すような $y$ 値の差の絶対値を $dy_{left}$（式 4.1）とする．同様に，右側で隣接する点 $p_{i+1}$ との $y$ 値の差の絶対値を $dy_{right}$（式 4.2）とする．また，図 4.4(c) に示すような，$p_i$ の左側に隣接する $k$ 個の点から求められる，$k-1$ 個の $y$ 値の差の絶対値 $dy_{i-k}, \ldots, dy_{i-1}$ の平均を $\overline{dy_{left}}$（式 4.3）とする．同様に，図 4.4(d) に示すように，右側に隣接する $k$ 個の点から求められる，$k-1$ 個の $y$ 値の差の絶対値 $dy_{i+1}, \ldots, dy_{i+(k-1)}$ の平均を，$\overline{dy_{right}}$（式 4.4）とする．

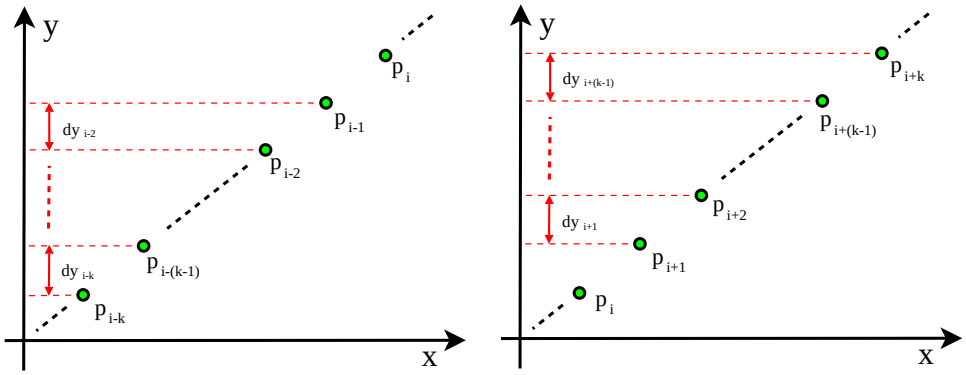
$$dy_{left} = |y_i - y_{i-1}|. \tag{4.1}$$

$$dy_{right} = |y_{i+1} - y_i|. \tag{4.2}$$

$$\overline{dy_{left}} = \frac{|y_{i-1} - y_{i-2}| + |y_{i-2} - y_{i-3}| + \cdots + |y_{i-(k-1)} - y_{i-k}|}{k-1}. \tag{4.3}$$




(a) $dy_i$ (b) $dy_{left}$ と $dy_{right}$




(c) $\overline{dy_{left}}$ (d) $\overline{dy_{right}}$

図 4.4: 異常値の判定基準

$$\overline{dy_{right}} = \frac{|y_{i+2} - y_{i+1}| + |y_{i+3} - y_{i+2}| + \cdots + |y_{i+k} - y_{i+(k-1)}|}{k-1}. \tag{4.4}$$

こうして求まった，$dy_{left}$ と $\overline{dy_{left}}$，$dy_{right}$ と $\overline{dy_{right}}$ を，何らかの基準を設けて比較することにより，$p_i$ が異常値か，そうでないかの判定が可能である．

本研究で，実際にこの異常値除去法を実装した際には，入力点数 50 に対し，$k$ の値を 5 とし，また $dy_{left}$ と 2 倍の $\overline{dy_{left}}$，$dy_{right}$ と 2 倍の $\overline{dy_{right}}$ をそれぞれ比較した，次の二式が共に成り立つ場合にのみ，$p_i$ を異常値と断定した．

$$dy_{left} \gneqq 2\,\overline{dy_{left}}, \tag{4.5}$$

$$dy_{right} \gneqq 2\,\overline{dy_{right}}. \tag{4.6}$$

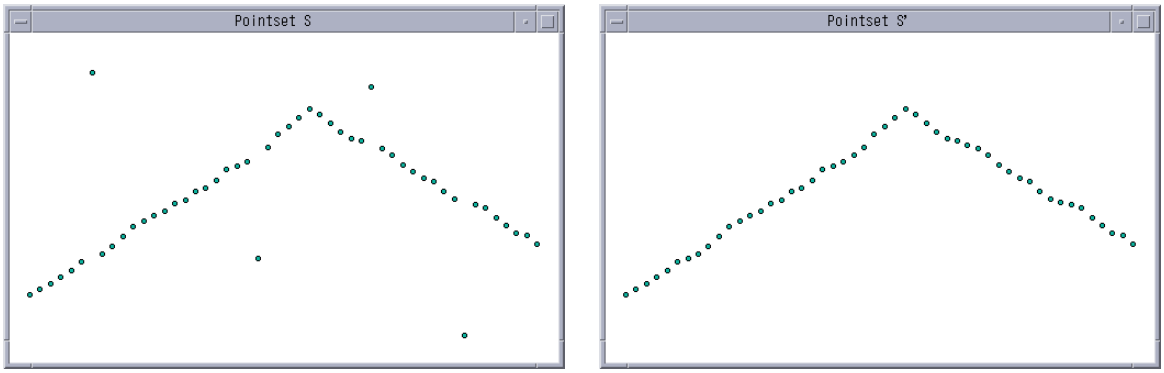
異常値と断定された点 $p_i$ に関しては次の式で表されるような，左右の点の $y$ 値の平均値を $y$ の値として持つような点に置き換える．

$$\frac{y_{i-1} + y_{i+1}}{2}. \tag{4.7}$$

そのようにして得られた点集合を入力の点集合 $S$ と区別して，点 $p'_i (i = 1, \ldots, n)$ からなる点集合 $S'$ とする．

以上の方法により，異常値を含む点集合 $S$ から異常値を除去した点集合 $S'$ を生成するのは，線形時間で可能である．

この，方法を実際に図 4.5(a) のような，異常値を含む点集合 $S$ に対して実行した結果が，図 4.5(b) の点集合 $S'$ である．異常値が連続して現れる場合や，頻繁に現れる場合を除いて，異常値の除去にはこの方法で十分であることがわかる．



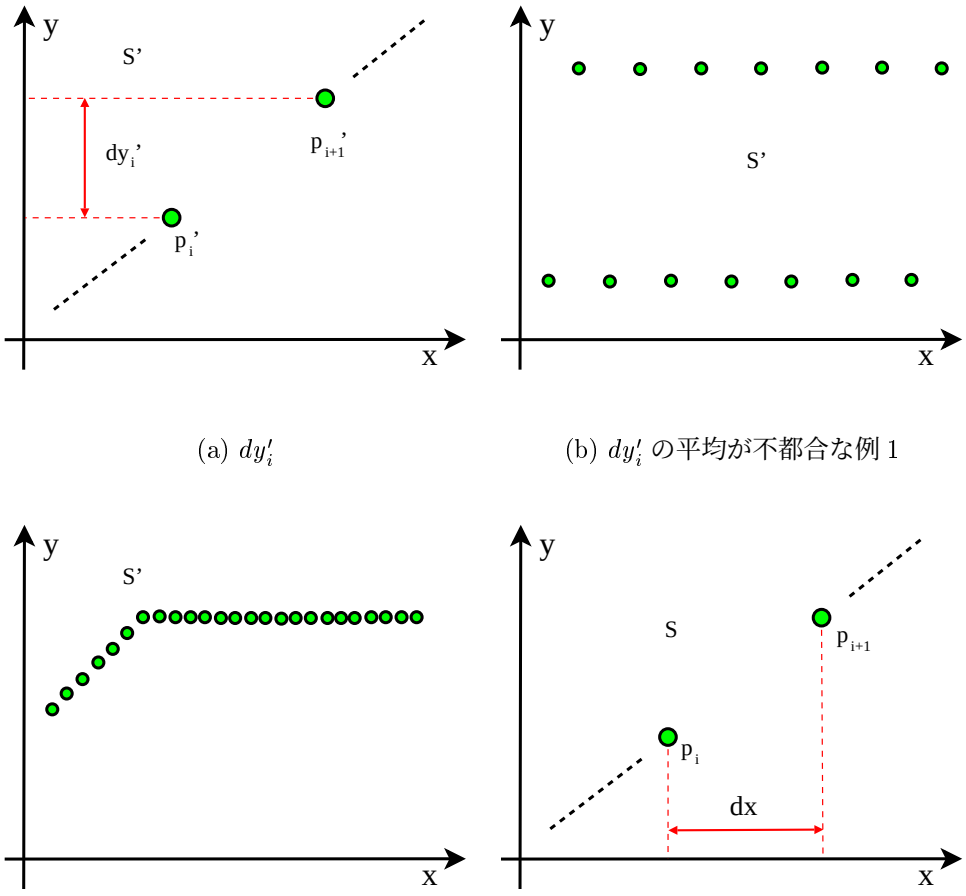

(a) 異常値を含む点集合 $S$　　(b) 実行結果の点集合 $S'$

図 4.5: 異常値除去の実行例

### 4.1.2 一連節折れ線近似性条件とその判定方法

本研究で提案するアルゴリズムでは一連節折れ線近似性条件として，3.2 節で紹介した，線形時間の $L_\infty$ ノルムを用いた折れ線近似の結果を用いる．具体的には，前節で紹介した異常値除去法を行った点集合 $S'$ に対して 3.2 節のアルゴリズムを適用し，得られた折れ線近似の屈折点が一つだった場合にのみ，入力の点集合 $S$ は一連節折れ線近似性条件を満たすとする．

ここで問題になるのが，$L_\infty$ ノルムの上限の値，つまり 3.2 節で言うところの誤差範囲 $w$ をどの様な値にするかである．これには，様々な値の決定法が考えられる．




(a) $dy_i'$　(b) $dy_i'$ の平均が不都合な例 1

(c) $dy_i'$ の平均が不都合な例 2　(d) $dx$

図 4.6: 誤差範囲 $w$ の決定

特に，図 4.6(a) に示すような点集合 $S'$ の各隣接点間の $y$ 座標値の差の絶対値 $dy_i{}' = |y_{i+1}{}' - y_i{}'|(i = 1, 2, \ldots, n-1)$ の次式で表される平均を用いる方法が妥当であるように

思われる．

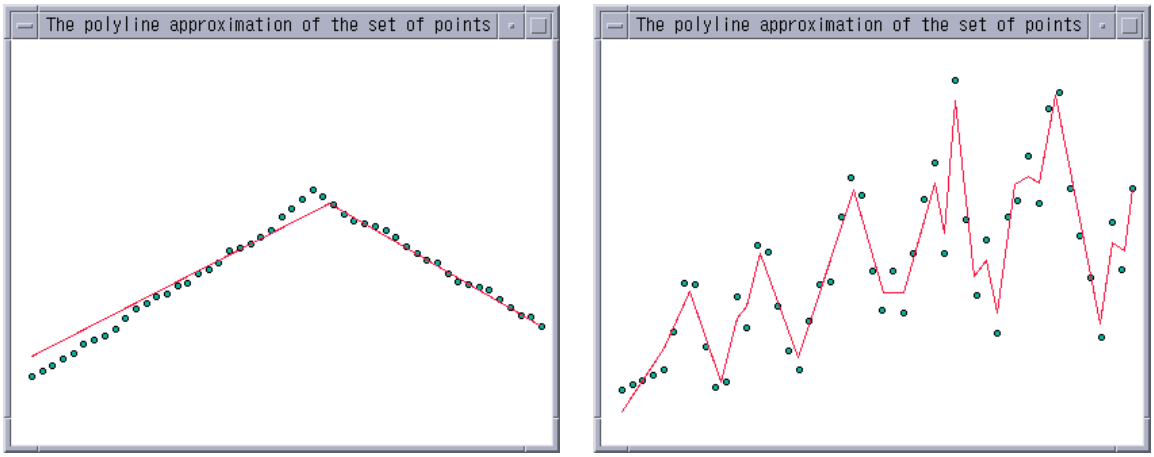
$$\frac{dy_1{}' + dy_2{}' + \cdots + dy_{n-1}{}'}{n-1}. \tag{4.8}$$

しかし，点集合が図 4.6(b) のような場合は，$dy_i{}'$ 平均も大きくなり明らかに一連節折れ線近似を用いるのが相応しくないにもかかわらず，一連節折れ線近似性を満たしてしまう．また，図 4.6(c) のように点列が $x$ 軸に平行に並んでいる場合は値が小さくなりすぎてしまい，一連節折れ線近似を用いるのに相応しい点列も除外してしまう可能性がある．

したがって点集合の形により変動する平均ではなく，何らかのある一定の値を用いる方が良いという結論に達した．そこで，本研究で実装するアルゴリズムでは，図 4.6(d) のような点集合全体で一定の値をもつ各点の $x$ 値の差 $dx$ の 2 倍を誤差範囲 $w$ として用いる事とした．これは，$dx$ がデータの変化をとらえるのに最適な間隔であるということも考慮した上での決定である．

以上のような，手法により点列の一連節折れ線近似性条件は線形時間で判定することが可能である．

上のような条件で実装したアルゴリズムを，実際に異常値を除去を施した点列に対して実行した結果を以下に示す．図 4.7(a) は一連節折れ線近似性条件を満たすと判定された点集合の例，図 4.7(b) は満たさないと判定された点集合の例である．図 4.7(b) に関しては，異常値が取り除かれていないように見えるが，これは各隣接点間の $y$ 座標値の差の絶対値がもとから大きかった為，平均を用いる異常値除去法の異常値判定につかまら無かった結果である．このように，規則性が無い点集合に関しては異常値の除去は行われず，一連節折れ線近似性の判定で省かれる．




(a) 条件を満たす点集合　　(b) 条件を満たさない点集合

図 4.7: 一連節折れ線近似性条件の判定結果

## 4.2 屈折点候補の絞り込み方法

既存の方法では，2章で紹介したように，屈折点候補の絞り込みは行わず，入力点の全てを屈折点候補として，それら全てに対し一連節折れ線近似を求め比較することにより，最適な一連節折れ線近似を求めている．

本研究では，特に実用性を重視することにより，近似対象である点集合を一連節折線近似を用いるのが相応しいもののみに絞った．そのことにより，一般的な点集合に比べ，折れ線近似のより最適に近い屈折点の位置を予想することが容易な点集合のみを近似対象にすることが可能になった．

そこで本節では，一連節折れ線近似性条件を満たした点集合に対し，屈折点候補を絞り込む方法を二通り紹介する．

一方は，一連節折れ線近似性条件に用いたのと同様に，3.2節で紹介した$L_\infty$ノルムを用いた線形時間の折れ線近似の結果を用いて屈折点候補を一つに絞り込む方法である．本論文では以降この方法を$L_\infty$法と呼ぶ．$L_\infty$法は4.2.1節で紹介する．

もう一方は，3.1で紹介した，線形時間の$L_1$直線近似の結果を用いて，屈折点候補を一つに絞り込む方法である．本論文では以降この方法を$L_1$法と呼ぶ．$L_1$法は4.2.2節で紹介する．

どちらの方法でも，より最適な折れ線近似を得るために，求まった屈折点候補のみでなく，周りのいくつかの点も屈折点候補とし，その定数個の屈折点候補に対し折れ線近似を求め，それらを比較することにより，より最適な折れ線近似を求める．

### 4.2.1 $L_\infty$法による屈折点の絞り込み

$L_\infty$法は，非常に単純である．4.1.2節の条件判定と同様，異常値を取り除いた点列$S'$に対し，3.2節で紹介した折線近似アルゴリズムを用い，その結果得られた点列の一連節折線近似の屈折点を，$L_1$一連節折れ線近似の屈折点候補として用いる．したがって，特に屈折点候補の絞り込みの為に行うことは無く，一連節折れ線近似性条件の判定の為に求めた$L_\infty$折れ線近似の結果をそのまま用いる事により，線形時間で屈折点候補が求まる．

図4.8に示すように，こうして求まった屈折点候補の左右3つずつ計6個の入力点を$L_\infty$法による屈折点候補とする．

### 4.2.2 $L_1$法による屈折点の絞り込み

異常値の除去を行い一連節折れ線近似性条件を満たすような点集合$S'$は，図4.9(a)，(b)に示すように，左端の点と右端の点に挟まれた領域の点集合が上下どちらかに突き出た形状であることが予想される．$L_1$法では，この様な点集合の形状的な特徴に注目し，折れ線近似の屈折点候補を一つに絞り込む．

3.1節でも述べたように，点集合に対する$L_1$直線近似は，点集合の2点を通り，かつ点

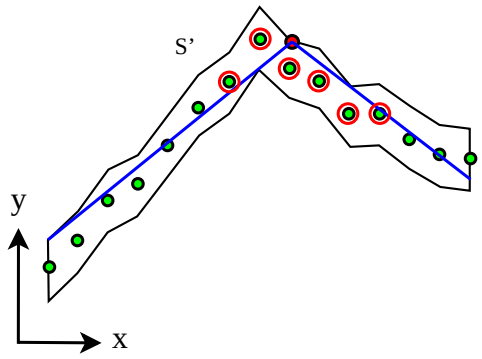
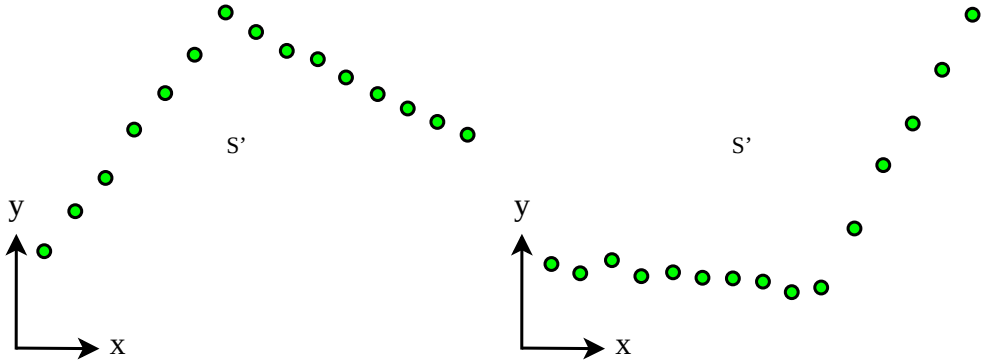



図 4.8: $L_\infty$ 法による候補点

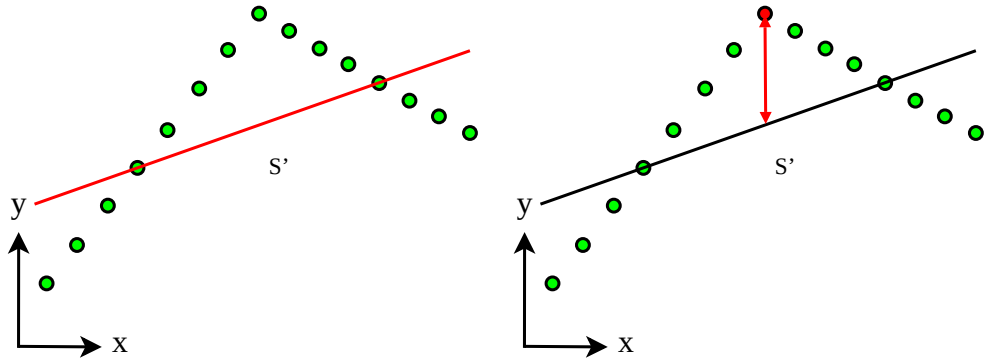



(a) 上に突き出した形状の $S'$　　(b) 下に突き出した形状の $S'$

図 4.9: 一連節折れ線近似性条件を満たす点列 $S'$ の形状

集合を近似直線の上下で二等分する．この $L_1$ 直線近似を一連節折れ線近似性条件を満たした点集合 $S'$ に用いると，図 4.10(a) に示すように点列を分割する．すると，図 4.10(b) に示すように，$L_1$ 近似直線によって分割された二つの半平面の内，両端の点を含まない側の半平面の点で近似直線との垂直距離が最も大きいものが，一連節折れ線近似の屈折点候補として，最も適した点として得られる．

以上のように，点集合 $S'$ に対する $L_1$ 直線近似の結果から，屈折点候補を一つに絞り込むことは，線形時間で可能である．実際の候補点としては，図 4.11 に示すように，上述の手法により得られた候補点の，前後 2 つずつ計 5 つを $L_1$ 一連節折れ線近似の屈折点候補とする．




(a) $L_1$ 直線近似による点列の分割　　(b) 屈折点候補の決定

図 4.10: $L_1$ 法による屈折点候補の絞り込み

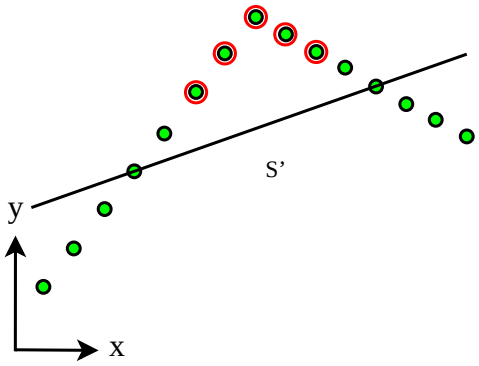



図 4.11: $L_1$ 法による候補点

## 4.3　折れ線の構成方法

折れ線の構成方法は，2 節で述べたものとほぼ同じである．ただし，本提案手法では折れ線の屈折点候補を定数個に絞り込んでいるの為，全探索を行う 2 節の方法では $O(n^2)$ の計算時間で最適な一連節折れ線金近似を求めるのに対し，$O(n)$ の計算時間で絞り込んだ定数個の一連節折れ線近似の内，最も最適に近いもの，または最適解を求めることができる．

$k$，$l$ を定数としたとき，絞り込んだ $k$ 個の屈折点候補が入力の点集合 $S$ の $l$ 番目の点から $l+k$ 番目の点だったとすると，本提案手法の折れ線構成のアルゴリズムは，2 節のアルゴリズムと同様に次のように表すことができる..

for: $q = l, \ldots, l+k$(分割点 $p_q$ を順次変更)

1. 点集合 $S$ を点 $p_q$ で左側点集合 $S_1$ と，右側点集合 $S_2$ に分割する．

2. 点集合 $S_1$，$S_2$ それぞれに対し $L_1$ 直線近似を用いて最適な近似直線を求める．
3. 求めた二本の直線からなる折れ線が式 2.8 の成立条件を満たしていればそれを最適な折れ線近似として出力する．
4. それ以外の場合は式 2.3 を制約条件として加えた上で，最適な折れ線近似を再び求める．

上記のアルゴリズムにより得られた，$k$ 個の一連節折れ線近似の $L_1$ ノルムの値を比較し，$L_1$ ノルムの値が最も小さな折れ線近似を求める．そうして求めた，$L_1$ 一連節折れ線近似が，本提案手法の出力の折れ線近似である．

## 4.4 提案手法のまとめ

ここでは，本提案手法の一連の流れをまとめる．提案手法は二種類あるが，屈折点候補の絞り込み法以外は同じであるので一つにまとめて示す．

入力として，$n$ 個の点 $p_1, p_2, \ldots, p_n$ からなる点集合 $S$ が与えられたとする．その時の本提案手法の手順を以下に示す．

1. 4.1.1 節で紹介した異常値除去法により，異常値を除いた点集合 $S'$ を構成する．
2. 点集合 $S'$ に対し 3.2 節で紹介した $L_\infty$ 折れ線近似を行うことにより，屈折点の数を求める．
3. 屈折点の数が 1 の場合は次の手順へ進む，屈折点の数が 1 以外の場合は終了する．
4. 4.2.1 節で紹介した $L_\infty$ 法かまたは，4.2.2 節で紹介した $L_1$ 法を用いて折れ線近似の屈折点候補を $k$ 個に絞り込む．ここで，$k$ は定数である．
5. 絞り込んだ $k$ 個の屈折点候補ごとに，点集合 $S$ の $L_1$ 折れ線近似を 4.3 節で紹介した，折れ線の構成法を用いて求める．
6. そうして求まった $k$ 個の一連節折れ線近似の内，$L_1$ ノルムの値が最も小さな折れ線を本提案手法の $L_1$ 一連節折れ線近似として出力する．

以上のような手順により，本提案手法における $L_1$ 一連節折れ線近似を得ることができる．

アルゴリズムの計算時間の評価は，それぞれの手順が線形時間で実行可能であるので，全体でも入力の点の数 $n$ に対する線形時間で実行可能である．

# 第5章　アルゴリズムの評価

ここでは，4 章で紹介したアルゴリズムを実装し実行した結果を 2 章で紹介した方法で求めた最適 $L_1$ 折れ線近似と比較する事により，本アルゴリズムの評価を行う．

## 5.1　実行結果と考察

50 個の点からなる，点集合に対して本論文で紹介したアルゴリズムを用いた結果を図 5.2，と図 5.3 にそれぞれ示す．図 5.2 のような点集合においては，$L_\infty$ 法をもちいた図 5.2(b)，$L_1$ 法を用いた図 5.2(c) ともに，図 5.2(a) の最適 $L_1$ 一連節折れ線近似と同じ結果を得る事が可能である．しかし，図 5.3 のような点集合においては，特に図 5.3(a) に示す $L_\infty$ 法を用いた一連節折れ線近似は，最適解の折れ線近似とはかなり違ったものとなっている．

これら二つの点集合の違いは屈折点の角度にある．図 5.2 の点集合は屈折角が大きく，図 5.3 の点集合は屈折角が小さい．この屈折角の大小により $L_\infty$ 法を用いた一連節折れ線近似はその精度を左右されることが分かる．

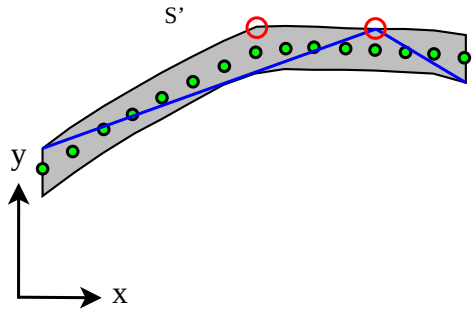


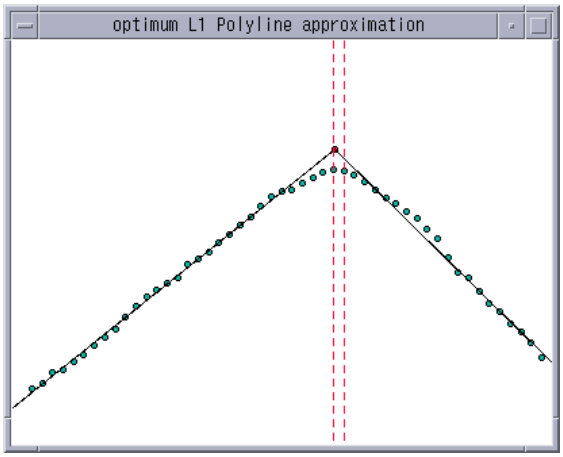

図 5.1: $L_\infty$ の欠点

この原因としては，$L_\infty$ 法で屈折点の候補を絞り込む際の，$L_\infty$ 折れ線近似の構成方法が考えられる．3.2 節で紹介したように，$L_\infty$ 折れ線近似を構成する際には，多角形内での辺からの可視領域を求め，その最大の範囲まで直線で結ぶことにより折れ線を構成する．したがって図 5.1 に示すように，屈折角度が小さい場合には折れ線の屈折点は，最適な位置から大きく右側へずれる可能性がある．特に点集合が単調である場合にそのようなこと

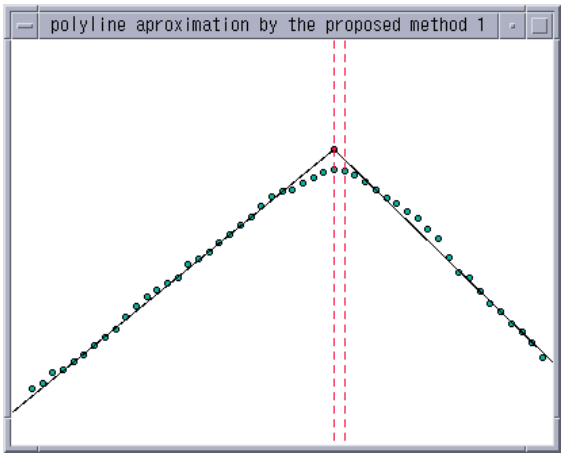
が起りやすい．これが $L_\infty$ 法を用いた一連節折れ線近似の欠点と言える．

$L_1$ 法に関しては，よほど屈折角が小さい場合を除き屈折点が一つの場合には，特に欠点らしきものが見つからない．また，屈折角が小さい場合は大抵一連節折れ線近似性を満たさずに，そん時点で実行が終了する為，一連節折れ線近似性を満たすような点集合に対してなら，最適解に非常に近い結果を得ることが可能である．$L_1$ 法の欠点としては，屈折点が一つの場合はかなり有効な方法であるが，屈折点数がそれ以上になるとその適用が非常に難しくなる点が挙げられる．

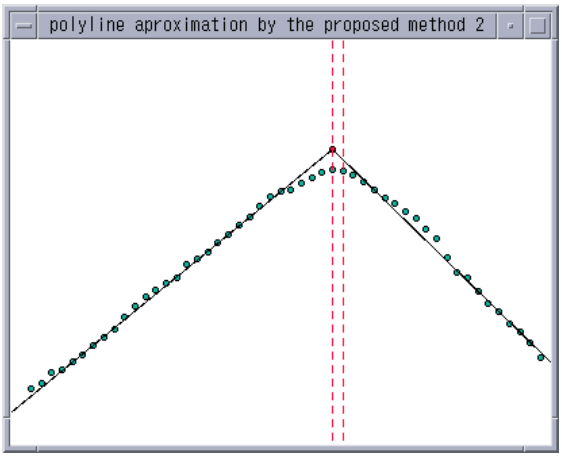
そういった，複数個の屈折点を持つ折れ線近似への適用を考えると，$L_\infty$ 法は比較的容易にそれらの問題に対応させることが可能である．しかし，この時はさらに最適解を得ることは難しくなることが予想される．




(a) 最適 $L_1$ 一連節折れ線近似

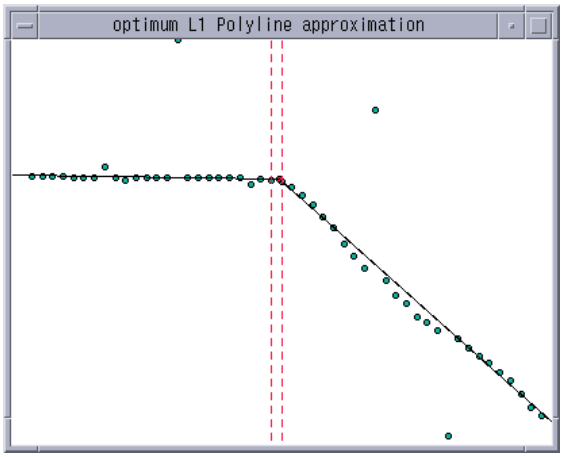



(b) $L_\infty$ 法による実行結果

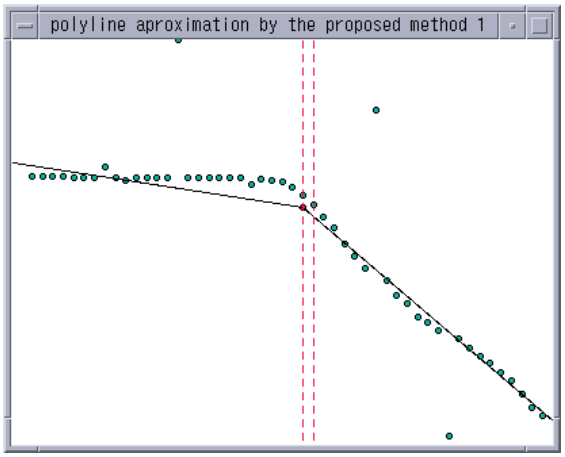



(c) $L_1$ 法による実行結果

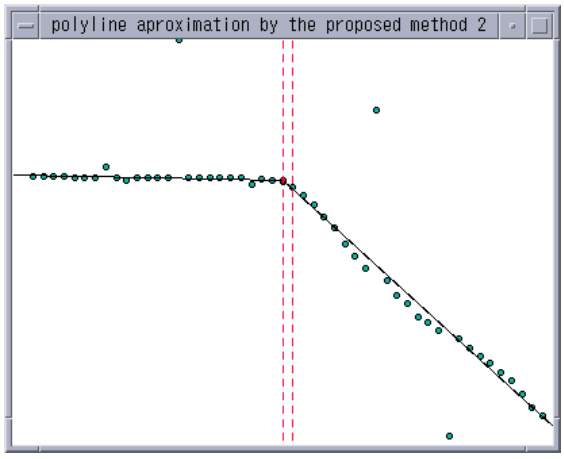
図 5.2: 提案手法の実行結果 1




(a) 最適 $L_1$ 一連節折れ線近似




(b) $L_\infty$ 法による実行結果




(c) $L_1$ 法による実行結果

図 5.3: 提案手法の実行結果 2

# 第6章　まとめ

## 6.1　結論

以上，本研究で考案した 2 つの $L_1$ 一連節折れ線近似アルゴリズムの紹介と，それらを実際に実装し，実行した結果の評価を行った．その結果として，$L_1$ 法を用いたアルゴリズムの方が，より最適解に近い折れ線近似を得ることが可能であることが分かった．また，もう一方の $L_\infty$ 法は精度の点では $L_1$ 法に劣るが，屈折点数を複数個に拡張した際の応用性の点では，より優れていると言える．

本研究で提案した，二つのアルゴリズムはともに最適解を得る保証は無いが，問題の入力のサイズに対する線形時間で，ある程度の精度をもつ点列に対する折れ線近似を求めることが可能である．

## 6.2　今後の課題

本研究で提案したアルゴリズムには，多くの問題点も存在する．特に一連節折れ線近似性の判定で誤差範囲一定の折れ線近似を用いたが，提案手法ではこの誤差範囲の値により点集合が一連節折れ線近似に相応しいかを判定している．したがって，この誤差範囲は非常に重要であり，ある程度の妥当な値を用いる必要がある．本研究でもこの点については考えたが特に決定的なものが見つからず，データのサンプリングの間隔はデータの値の変化をとらえるのに最適な値である，という仮定のもとでそれを基準として用いる事とした．この点については，まだ考慮の余地があると思われる．

また，この本提案手法は異常値の除去ができる事が前提で成り立っている．例えば，図 4.5(a) で示したような，ごく離れた位置に，ごくわずかの異常値しかない場合なら本研究で提案した異常値の除去法で十分であるが，連続で頻繁に異常値が現れるようなデータに対しては，本提案手法の適用は不可能である．その点に関しても，まだ改善の余地があると思われる．

本手法は将来的には屈折点数を複数個に増やした場合に適用する事も考えて考案した．$L_1$ 法に関してはそういった問題への適用は困難であるが，$L_\infty$ 法に関しては比較的容易に複数の屈折点をもつ折れ線近似問題に拡張できそうである．したがって，$L_\infty$ 折れ線近似の結果を用いて屈折点の絞り込みを用い，$k$ 連節の $L_1$ 折れ線近似を求めることのできるアルゴリズムの開発が今後の課題として挙げられる．

# 謝辞

浅野哲夫教授，上原隆平助教授，元木光雄助手，Arijit Bishunu 助手をはじめ情報三棟六階を主な研究場所とする皆さんには，ゼミや日頃の会話を通して多くの有効な助言を頂き，また様々な場面で助けて頂くと同時にご迷惑もお掛けしたと思います．この場を借りてお礼申し上げます．特に指導教官の浅野哲夫教授には研究の方向性や，その進め方について数々の大切な意見やアドバイスを頂きました．その助言があってはじめて，研究を一つの論文としてまとめることができました．本当に感謝に堪えません．

機械科出身の私が北陸先端科学技術大学院大学の情報科で二年間頑張って来れたのも，そうした研究室の皆さんの助けと同時に友人達が日頃から気分転換に付き合いつつ励ましてくれたからこそだと思います．そうした友人達に感謝します．

最後に，これまで多くの人生の場面で，私を支え見守ってくれた両親，困難な時に励ましてくれた妹，この三人の家族に深く感謝します．

# 参考文献


[1] H. Imai, K. Kato, and P. Yamamoto: "A linear-time alogorithm for linear $L_1$ approximation of points", Algorithmica, 4(1), pp.77–96, 1989.

[2] 伊理 正夫 監修，腰塚 武志 編集，今井 浩 著: "第２版 計算幾何学と地理情報処理", 共立出版株式会社，p.121-124, 1993.

[3] 今井 浩: "計算幾何学における算法の研究", 東京大学大学院工学研究科情報工学専門課程学位請求論文，pp.238-273, 1986.

[4] ボリス アロノフ, 浅野 哲夫, 加藤 直樹, クルト メルホルン, 徳山 豪: "距離最小化基準による点集合の折れ線近似", 情報処理学会 研究報告, 2004-AL-96(8),pp.51–58, 2004.

[5] M.T.Goodrich: "Efficient Piecewise-Linear Function Approximation Using the Uniform Metric", Descrete Comput Geom, 14:445-462(1995)

[6] S.L.Hakimi, and E.F.Schmeichel: "Fitting Polygonal Functions to a Set of Points in the Plane", Graphical Models and Image Processing, Vol.53, No.2, March, pp.132-136, 1991.